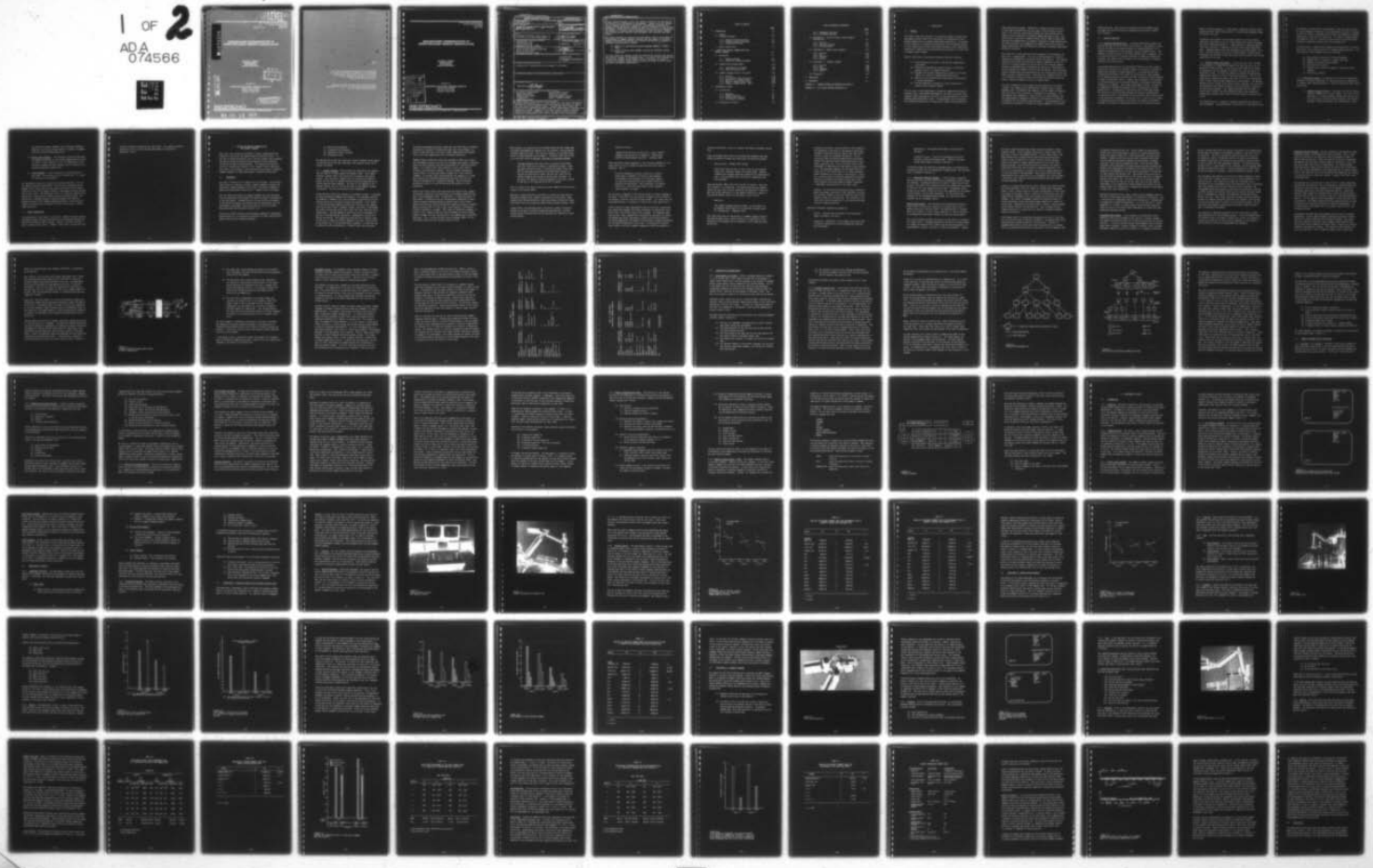
AD-A074 566 PERCEPTRONICS INC WOODLAND HILLS CALIF F/G 5/8
MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION--ETC(U)
MAR 79 W H CROOKS, E SHAKET, Y CHU N00014-76-C-0603
UNCLASSIFIED PATR-1034-79-3 NL

1 OF 2
ADA 074566




1 OF 2
AD A
074566
1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART

(12) LEVEL II

Technical Report PATR-1034-79-3
Contract N00014-78-C-0683
NR 196-140
March 1979

AD A057961

AD A074566

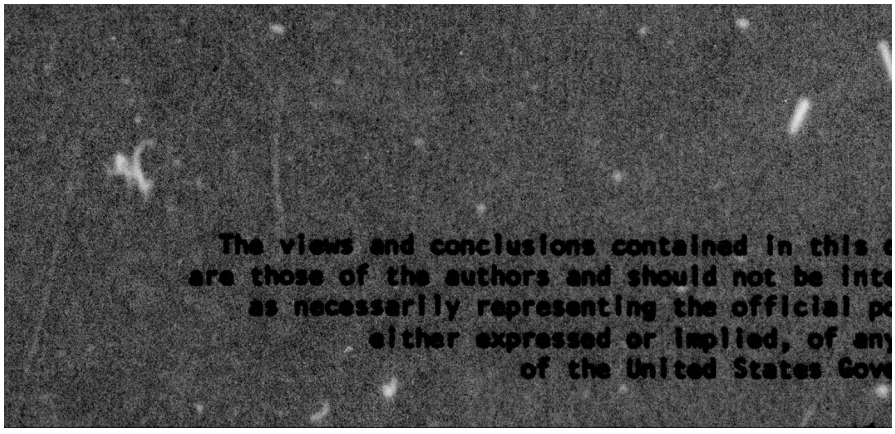
MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION

WILLIAM H. CROOKS
EFRAIM SHAKET
YEE-YEEN CHU
YORAM ALPEROVITCH

Prepared For:

ENGINEERING PSYCHOLOGY PROGRAMS (CODE 455)
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

PERCEPTRONICS
6271 VARIEL AVENUE • WOODLAND HILLS • CALIFORNIA 91367 • PHONE (213) 884-7470


The views and conclusions contained in this
are those of the authors and should not be int
as necessarily representing the official p
either expressed or implied, of an
of the United States Gov

Technical Report PATR-1034-79-3
Contract N00014-76-C-0603
NR 196-140
March 1979

MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION

WILLIAM H. CROOKS
EFRAIM SHAKET
YEE-YEEN CHU
YORAM ALPEROVITCH

Prepared For:

ENGINEERING PSYCHOLOGY PROGRAMS (CODE 455)
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

PERCEPTRONICS
6271 VARIEL AVENUE • WOODLAND HILLS • CALIFORNIA 91367 • PHONE (213) 884-7470

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: PATR-1034-79-3	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION.	5. TYPE OF REPORT & PERIOD COVERED: Technical Report, 2 Feb 1978 to 1 Feb 1979
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): William H. /Crooks, Efraim /Shaket, Yee-Yeen /Chu Yoram /Alperovitch	8. CONTRACT OR GRANT NUMBER(s): N00014-76-C-0603
9. PERFORMING ORGANIZATION NAME AND ADDRESS: PERCEPTRONICS, INC. 6271 Variel Avenue Woodland Hills, CA. 91367	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR196-140 167
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research (Code 455) 800 N. Quincy Street Arlington, Virginia 22217	12. REPORT DATE: March 1979
	13. NUMBER OF PAGES: 178
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Unclassified - Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved by

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Man-Machine Communication
Remote Manipulators
Computer-Aided Control
Teleoperators
Supervisory Control
Resolved Motion Control
Hierarchical Planning
Shared Man-Computer Control
Symbolic Command Language

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Automated Remote Manipulation is a prime example of a new type of man-machine interaction in which the human operator must supervise and control a complex and often adaptive man-computer system. Computerized control offers the possibilities of improved performance times and reduced operator workloads with teleoperator systems. Computers can be used at various levels of control, ranging from control augmentation, where the computer performs difficult coordinate transformations which simplify operator control requirements,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

20.

through complete autonomy in which the computer performs all of the required activities with no intervention by the operator. However, with the introduction of computer-based control techniques, the communication between the operator and the teleoperator becomes an important determinant of work system performance. Rather than controlling directly every action of the manipulator, the operator of a computer-controlled manipulator plans the tasks, commands goal-directed actions, monitors task performance, and intervenes when appropriate.

This report describes an analytical and experimental study to investigate the effectiveness of command language structures and the methods for providing feedback information through the use of sensors and displays. A sequence of experiments was performed to examine:

(1) Benefits of user-defined variable language commands in complex tasks.

(2) Level of machine state feedback required for different viewing conditions.

The study showed that computer aiding can significantly decrease task performance time for a number of teleoperator tasks. The results also indicated that if high-level computer aiding schemes are to be effective, the design of interface and feedback display must be carefully performed to achieve simple and natural man-machine communication.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

1. INTRODUCTION

1.1 Summary

This report covers the third year of a four-year program of research and development directed toward the investigation and optimization of man-machine communication in computer-aided remote manipulation. The purpose of this program is to determine experimentally the relationships between critical communication factors and system performance, and to develop and demonstrate a communication design methodology applicable to a broad class of remotely manned systems.

Specific objectives of the research program include the following:

(1) Perform theoretical analysis of man-machine communication requirements.
(2) Establish an experimental system for study of shared man-computer control of a remote manipulator.
(3) Implement and evaluate communications systems for efficient control of a variety of remote manipulation tasks.
(4) Identify critical system factors and establish relationship to system performance.
(5) Provide guidelines for design for future autonomous and adaptive remotely-manned systems.

The first year's study established a theoretical communications based on procedural nets, and examined experimentally the effect of several basic computer aiding techniques on the ability of trained operators to perform selected remote manipulation tasks. The experimental results indicated that computer-aided control can significantly improve the performance of

remote manipulation systems. Specifically, computer aiding in the form of real-time transformation from joint angle to resolved motion control (RMC) of the end point significantly reduces the time required and the number of errors committed in performing remote manipulation tasks. Computer aiding in the form of automatic motion control (AMC) to specifiable locations also demonstrated its potential usefulness in remote manipulation. The training results also suggested that computer aiding can be used to reduce significantly the time required for personnel to become accomplished manipulator operators.

The second year's study emphasized the development of the procedural net model into a language model. This led to the development of a specific language and corresponding input/output devices and protocol. The language, through the definition and execution of symbolic commands, provides the user with a flexible mechanism to define and use task oriented commands. Extensive reprogramming was performed during the year to incorporate all the new features and feedback facilities into the system. The experimental results indicated that pre-established chains can significantly reduce task time to perform a repetitive task. These data also suggested that the benefit of chaining increases with practice and that the chaining capability helps to reduce the number of errors in a complex task.

This year's theoretical work provided (a) the extension of the command language, with command chains embedded within command chains allowing expanded levels of predefined task commands, and (b) feedback information about the state of the system that is presented to the operator. The experimental phase examined (1) visual display, (2) state feedback, and (3) command mode as independent variables in the evaluation of operator performance with the computer-aided manipulator. The experimental results indicated the usefulness of variable and chained commands in an indirect

viewing situation. The results also suggested that the feedback format and update rate should be carefully designed and controlled for better user acceptance.

1.2 Technical Approach

1.2.1 Computer-Aided Manipulation. In present teleoperator systems, the operator performs all of the control tasks manually. One method to improve the slow and often error-prone performance of remote manipulators is to augment or automate most of the operator's control responsibilities. This approach allocates functions to the human operator and to a computer. Such an allocation retains the favorable attributes of human intelligence and foresight, and combines these attributes with the advantages of automatic, computer-controlled operation.

As we extend shared control of a manipulation system to the full range of capabilities afforded by the computer element, the question of man-machine communication becomes of primary importance. In any manipulation task, the operator must observe the actions of the manipulator, make judgments of the commands necessary to perform the task, and carry out those judgments in terms of manipulator control. However, the involvement of the operator in performing these tasks is strongly affected by the degree of assistance provided by the computer. When using unaided manual control, the question of communication between the operator control and observation of every machine action. Introduction of supervisory control techniques changes the character of the relationship between the operator and manipulator. In supervisory control, the operator not only provides direct analog control of the manipulator's movements, but he must also (1) select any of a number of computer-associated functions, (2) monitor the progress of automated routines, (3) be able to resume manual control, and (4) know what control

mode is currently operating. A wide range of communication modes, encompassing more than simple analog control, is required when augmented remote manipulators are used.

Accordingly, the approach in the present program is to focus research attention on the general rules for constructing special-purpose languages. The case of remote manipulation is a good example of a bounded communications area, one which is important in its own right to Navy operational goals. Language elements have been implemented at the man-computer interface. The relationship of variation in these elements to total system performance has provided the data upon which practical human factors design guidelines will be based.

1.2.2 Command Language Development. The previous year's work included the development of a communication model which forms the basis for the man-machine language design. A hierarchical model based on the concept of procedural nets was developed to represent the planning process of manipulator action. The model was adapted to represent the man-machine communication process as a step in this planning process. That is, the human operator develops the global task into a plan at some intermediate level of detail which he then communicates to the computer. The computer develops the plan further to the level of detail necessary for controlling the manipulator and proceeds to monitor the plan execution via position and force-sensor feedback. The language developed from this model is hierarchical, flexible, and task-oriented allowing close cooperation between man and machine with a mixed initiative protocol of control allocation.

The procedural net is a conceptual framework that models the process of plan development. The problem domain is described as a hierarchy of tasks

and subtasks at various levels of abstraction. Each task node consists of a goal statement of what has to be accomplished by the task, and object on which the action is performed, and an action--the sequence of subtasks expressed at a lower level of abstraction. The specific sequence needed to accomplish a task is a function of both the state of the environment and what is requested at higher levels of the global task.

Using this model, a communication language was designed which facilitates hierarchical, task-oriented control of a computer-aided manipulator. The main features of the language include the following:

(1) Task-oriented, hierarchical structured commands.
(2) Command chains as concepts at various task levels.
(3) Sentence-structured keyboard command.
(4) Mixed-initiative control.
(5) Intermixing of analogic and symbolic, complex and simple commands.
(6) Hierarchical feedback.

1.2.3 Experimental Program. The objective of the overall experimental program is to evaluate techniques for improved man-machine communication in computer-aided remote manipulation. The major dimensions under investigation were:

(1) Command Language Structure -- the manner by which the operator transmits commands to the remote element. The levels of this dimension are (1) Manual Commands, whereby the operator actuates analogic and individual control sequences, (2) Variable Commands, in which the operator uses defined points, paths

and individual symbolic commands, and (3) Chained (Automatic) Commands, in which the operator defines a sequence of commands which are then performed automatically.

(2) Machine State Feedback -- the information concerning the queue of commands established for the chained command mode of control. The levels of this dimension are (1) full delineation of the individual commands and their sequence, and (2) display of the currently operational command only.

(3) Visual Feedback -- visual observation of the manipulator in work space. The two levels of this dimension are: (1) normal TV viewing and (2) degraded TV.

The experimental studies are conducted using the Perceptronics' facility for computer-controlled manipulation. This facility, developed and programmed in the previous years of the program, provides all the necessary capabilities for testing language features, input/output protocol and feedback display levels. The manipulator itself is a dexterous, hydraulically-powered unit, combining quick response accuracy and high strength. The manipulator is commanded through a dedicated keyboard and joysticks, designed according to the command language guidelines, a CRT display for feedback of machine state information, and two video displays to permit remote operations.

1.3 Report Organization

The organization of this report is as follows: Chapter 2 reviews selected background concepts and the current design of our Shared Man-Computer Communication and Control (SMC^3) Language. Chapter 3 describes the experimental program and the results. Chapter 4 closes with a discussion of the

research findings and guidelines for application. The computer controlled manipulator facility and the software development are described in Appendices A and B.

2. A SHARED MAN-COMPUTER COMMUNICATION AND CONTROL LANGUAGE

Man's tools have increased continuously in power, sophistication and complexity, creating an ever greater need for effective control. In the case of remote manipulators, one of the most complex tools assembled, the control problem is especially acute because of their dexterity, flexibility and general purposefulness. This chapter will provide a historical overview and discussion of our research efforts addressing the various issues relevant to the understanding and design of effective command languages for remote manipulators in general purpose tasks.

2.1 Background

The areas of research we will address are the following: task analysis, human factors in interactive systems, command languages in general and manipulator command languages in particular, and previous attempts at modeling the man-machine interaction in a command control environment.

Both the theory and practice of command languages have been developed since the beginning of the punched card control systems at the beginning of the 20th century. Although the kind of tasks and the environments of the controlled systems are different from the manipulator tasks and environments, some general ideas and principles can be transferred. These principles are discussed in Section 2.1.1.

Section 2.1.2 covers previous work on command languages for manipulators more specifically. To structure the discussion, we have classified this general area into four separate approaches:

(1) Machine shop paradigm.
(2) Programming shop paradigm.
(3) Supervisory control paradigm.
(4) Autonomous control.

Our approach falls under the supervisory control paradigm, and by comparing and contrasting the other approaches, the important issues can be brought into focus.

2.1.1 Command Languages. Shortly after the invention of the computer, came the idea of a stored program (Von Neumann, 1946). This concept produced a quantum jump in the flexibility and speed of the control process. Now, the control commands could be processed at electronic speeds, and the order of command execution could be changed during run time by jump and branch commands. The level of the commands in the program, however, was still at the low level of the computer internal instructions. This was called binery level programming.

During the early fifties the concept of an assembler evolved. It provided an easier man-machine communication language. The programmer specified his commands in mnemonic codes, and the burden of translation to binary code was placed on a translator program--the assembler. This concept of relieving the human operator by using more and more of the computer itself was a major trend in computer science; since then, it has become the central driving idea within the science (see Treu, 1975) and can be stated simply: Use the computer to provide an easier environment for the user in the man-machine interaction. Subsequently, the idea of a "Macro" has evolved. This capability allows the user to define, label and then call a group of primitive commands as one unit. When carried to the extreme (as in the multipass Macro assembler) it allows the user to define his tasks hierarchically. Dijkstra (1972), and the recent

structured programming discipline, argue that such hierarchical, systematic structuring is essential for producing large, error free programs. We agree, and add that this is also so in other communication environments, such as real time communication with a dexterous manipulator.

FORTRAN (Formula Translator), which was developed in 1954, was a tremendous achievement because it was the first language (and compiler) developed around the data structures and control mechanisms of a specified problem domain. It is what we call "task oriented," in this case, solving scientific and statistical problems. The APL language, developed by Iverson (1962) is an even more dramatic example of a "task oriented" language. In the book cited, Iverson develops the language as a systematic formalism for algorithm representation and only later was this formalism implemented as a programming language. Much of the power, naturalness and ease of use of APL has been attributed to its task orientation.

Falkoff and Iverson (1973), while describing the design of APL, suggest additionally, simplicity and practicality as the general guidelines they used in the development of APL. In particular "...Simplicity enters in four guises: Uniformity--rules are few and simple; generality--a small number of general functions provide as special cases a host of more specialized functions; familiarity-familiar symbols and usage are adopted whenever possible (this is also in part "task oriented"); and brevity--economy of expression is sought." APL, being an interactive language, is similar in this respect to the real time requirement of the manipulator command language, and thus consciousness of the commands was an important constraint. This is unlike the more verbose languages--COBOL, PL1 and even PASCAL--where the time constraint does not exist.

More recently, as interactive on-line systems became the more common mode of software development and computer use, the interest in the man-machine interface aspect has grown. This interest, however, has not been disseminated to system designers and developers. As Bennett (1972) has written in an excellent review on The User--Interface in Interactive Systems,

> "Software designers have been justly criticised for providing tools that force users to behave in nonproductive modes. While designers have been correct in forseeing new modes, either they did not anticipate new patterns accurately (poor design), or they did not effectively transfer the the user's mind the conceptual framework that guided the design (poor training). In any event, the impact on system performance of the user's concept of the tool is too important to be left to chance."

This is related to the idea of making the system compatible with the user's concept of the problem domain.

Additional relevant research can be found in relation to other areas of man-machine communication, especially man-machine "conversation" about graphical data (Foley and Wallace, 1975), man-display interaction, (Engel and Granada, 1975) and interactive administrative systems (Kennedy, 1974).

Kennedy (1974), who developed several interactive systems for hospital systems to be used by relatively unsophisticated operators, outlined several design principles that are also applicable to interactive control of manipulators.

Sentence Structure:

> "Communication should be carried out in a terse 'Natural' language, avoiding the use of mnemonics. Abbreviation should be allowed whenever possible." (Kennedy, 1974)

Foley and Wallace (1975) suggested, in fact, that the commands in a language should be task-or-concept-oriented and should have sentence structure. That is:

> An action language is *sentence structured* if, within a given phase or subdomain of discourse, each complete user-thought can be expressed in a continuous sequence of input device manipulations with standard patterns of beginning and termination. Upon termination, the machine returns to a state from which similar action sequences, other sentences, can begin."

The structure is enhanced if the verb-noun format of natural language is utilitzed. Treu (1975) calls the essence of command "action primitive" and suggests a specific structure to each command: "(1) action verb; (2) action qualifier(s); (3) object(s) of action; (4) object qualifier(s)."

Ferrell (1973) has shown experimentally that an artificial, constrained and standardized language facilitates performance in manipulative tasks better than a free-format, English-like language. The advantage comes from the fact, that, although entire manipulation task goals are readily and perhaps most easily described by a person using ordinary English, the complex geometrical and temporal configurations of objects and motions are not readily formulated in such a way. A structured but flexible, task-oriented, artificial feedback language can enable the operator to

work more efficiently, as well as simplify the process of machine translation.

Engel and Granada (1975) concur with Kennedy that mnemonics and codes burden the operator's memory and cause errors and should be avoided.

Error Recovery: Kennedy (1974) states:

> "Each entry should be short so that errors can be corrected simply and a reasonable tempo can be established." However, "verbose error messages should be avoided for the sophisticated user" ...and he should have a facility for suppressing long error messages."

Engel and Granada (1975) state this principle as follows: "Any error can be undone" in a natural way. The system accepting user commands should have easy error recovery capability including both a display of what the system understood to be the command entered and a memory of the previous machine state to be recovered in case of error.

Simplicity:

> "The command language should be simple, and the behavior of the computer should appear to be logically consistent under all circumstances." (Kennedy, 1974)

Treu (1975) thus calls for the design of a command language "based on the required mental work," which he defines as a combination of the complexity of the commands and the complexity of the system perceived by the user.

Conceptual Continuity: Foley and Wallace (1975) emphasize the importance of continuity in the operator sensory and conceptual interaction with the system, what they called "tactile, visual and conceptual continuity." *Tactile continuity* refers to natural grouping and flow of motion required for operating tactile input devices such as keyboards, joysticks, etc. *Visual continuity* refers to the arrangement of information, so that within a given sentence (i.e., one conceptual command), the eye should focus on a single area of the control panel or in a more continuous manner throughout the expression of the sentence. *Contextual continuity* refers to providing immediately perceivable responses to reinforce the effect of every step in the action sequence and giving standard feedback information in dedicated, fixed positions in the visual field.

Feedback: In dealing with man-machine display interfaces, Engel and Granada (1975) point to the importance of providing continuous feedback to the user about the state of the computer system he is dealing with. The information alleviates the frustration generated in the operator when dealing with a complex black box.

Additional principles indicated by Kennedy are:

Control: "Control over all aspects of the system must appear to belong to the user."

Redundancy: "Redundancy in the dialogue should be avoided or reduced, especially as the user becomes more familiar with the system."

Adaptability: "The system should adapt to the ability of the user."

Communication Rate: "The rate of exchange must be within the user's stress-free working range. Control of the rate should always appear to belong to the user."

A conscious attempt was made during language design to incorporate the relevant principles into the manipulator language design. The incorporation is discussed in Section 2.3.

2.1.2 Manipulator Command Languages. In this section, we describe previous work which was done specifically on manipulator control systems and languages. Although manipulators have been in regular use for over 30 years, relatively little research has been done addressing the problem of effective real time communication with them. Teleoperator control systems can be roughly classed into one of four approaches to the communication problem. We call these approaches: (1) the machine shop paradigm, (2) the programming shop paradigm, (3) supervisory control paradigm, and (4) autonomous control.

Machine Shop Paradigm. This approach views the man-machine system as being similar to the control situation in a mcahine shop, such as a mechanic operating a precision lathe. The machining tool is securely held; and its motion, in two or three dimensions, is deftly controlled by corresponding analog inputs entered via precision control wheels.

This control approach has been carried over to the control of teleoperators almost intact. Although teleoperators may have six or more degrees of freedom, the operator still controls each of them individually, via potentiometers, switches, joysticks, or other such actuators. Johnsen

and Corliss (1967) describe many systems using this approach in their extensive survey. In most systems, the operator's direct view of the work area is used as the feedback channel. In such direct control, the human operator has to produce several smooth and coordinated analog signals, a task in which he is, at best, slow and inexact, getting worse as the number of signals needed increases or his visual field becomes obscured. With sufficient training, a mechanic can control a lathe quite effectively, but to control a dexterous manipulator with five, six or seven degrees of freedom, the limits of human capabilities for real time control are approached. Pesch, et al (1970) have shown experimentally that a more than 10 to 1 performance ratio exists between a free driver and a remotely controlled manipulator when both perform the same underwater task.

Viewing as an analog input device by itself, master-slave control arrangements were developed in the later 1940's to overcome these limitations (Goertz, 1954). Master-slaves are bilateral teleoperators in which forces and torques at the master control, located out of the hostile area, are proportionally reproduced at the slave station and vice versa for force feedback. This control arrangement replaces the conscious control of several manipulator joints with natural human capability to control his arm and hand. Operation of the master-slaves is natural, and the operator easily projects himself into the work area. They are among the most common teleoperators in operational use, mainly in radioactive material handling laboratories.

Direct human control in the machine shop approach not only is very slow, but also severely taxes an operator's attention. The operator must direct the manipulator through the details of every motion. Attention is demanded particularly in those cases where the operator has to overcome built-in system deficiencies such as time delays, no force feedback, high

or unnatural inertial forces, etc. Ferrell (1965) has shown experimentally that time delays beyond a fraction of a second substantially disrupt the control process for a simple manipulator with only visual feedback. Subjects avoid the slow and erratic movements that delays tend to induce by consistently adopting the move-and-wait strategy, resulting in very high task completion times. Hill and Sword (1973) conducted similar experiments verifying the move-and-wait phenomena and showed similar degredation in performance with feedback delays; visual obstruction and degredation of visual quality were added to disrupt the TV signal coming from the work area. These difficulties can be partially alleviated by delegating part of the control function to a computer located at the remote site. The control loop will be closed through the computer which will monitor the low level tasks, and the capacity requirements of the communication channel between the operator and the remote system can be reduced with a commensurate reduction of bad effects of delays and visual distortions.

Additional application areas, where a master-slave arrangement cannot be utilized, are those where the operator is restricted in his motions. This is the case with a handicapped person whose missing arm is replaced with an automatically controlled manipulator. It is also the case in a small submersible or in space in a space capsule, where available space precludes the use of a full motion master manipulator to control the slave manipulator operating outside.

Programming Shop Pardigm. In this approach, the manipulator control problem is taken to be analogous to the control of a digital computer. The computer can perform a small set of primitive actions on its internal registers. It can also read and execute these commands from an internal coded form--a program. To cause a computer to perform a task, the human programmer expresses the task in full detail as a sequence of primitive

actions, stating specifically the actions to be taken in all eventualities. He then feeds his program to the computer, debuggs it, and turns it loose for execution. This basic approach was again carried over to the area of manipulator control. Manipulator motions are broken into primitive and complex trajectories are achieved by using control stuctures similar to those in other real-time programming languages. The system described by Wang (1976) is a good example of this approach, with the language ALPHA designed for manipulator control. Tedious programming and tight control of the environment are necessary to make the manipulator perform a task.

The programming shop paradigm is the one most commonly adopted for industrial robots and in number, exceeds by far, all other approaches to manipulator control. For industrial use where the environment can be controlled and the reprogramming time of the manipulator task is considered short when compared with specially designed and built machinery, the reprogrammed manipulator provides a cost effective alternative. Nevin et al (1973) and Nevin and Whitney (1978) describe another successful system which is programmed by "showing" the manipulator through the detailed motions of a task. Then, using a specially designed, passively compliant gripper, the computer-controlled manipulator can assemble a small production electric pump, made of 17 parts, in less than two minutes. All the parts must come, however, through specially designed feeding stations--again a predesigned environment.

This approach is completely unsuitable for real time control of underwater manipulators for the following reasons: (1) the environment cannot be controlled, and is not known in advance, and in fact, can change during the task execution; (2) tasks are not repeatable in their details; and (4) there is no time for preprogramming and debugging.

Supervisory Control Paradigm. The third approach to the problem of remote manipulator control is the supervisory control paradigm. Here, a supervisor sets the goals of the operation, gives the sequence of tasks in general terms and monitors loosely the progress as the task is performed. The operator does the actual motions, adapts to the details of the situation and corrects small errors as they develop. For larger problems, which the operator cannot solve by himself, he reports back to the supervisor, who is more knowledgeable, more experienced and is a better problem solver. The supervisor will solve the problem and communicate the solution. This solution may be an additional subtask, a change in the plan or even a new global goal.

This approach can be carried over to the control of a remote manipulator. The human has the supervisor role of setting goals, outlining plans, monitoring and solving top level problems, and the computer is the semi-autonomous controller of the actual manipulator motions. Ferrell and Sheridan (1967) suggest the following cases in which the human operator will not be able to act as if he were on the spot, doing the manipulations himself: "(1) The operator may be handicapped, or other tasks may share his attention. (2) Signal transmission may be affected by noise or power limitations (thus limiting the amount of information transmitted). (3) If the distance to the remote site is very great, there will be a long transmission delay."

Experiments have shown that these problems actually result in degraded performance. Ferrell (1965) conducted laboratory experiments in human control of a simple manipulator with transmission delays. The experiments have shown that with only visual feedback, human subjects avoid the slow and erratic movements that delays tend to induce by consistently adopting a move-and-wait strategy. Task completion time increases linearly with the delay. Ferrell (1966) has shown similar results when

delays are introduced when force feedback information is transmitted to the operator.

More recently, Hill and Sword (1973) report experiments with a system that adds both delays and channel noise to the tactile, visual and distance feedback information presented to the user. The experiments verify the wait-and-move strategy adopted by the user in the delay experiments. They have also shown the degradation of performance in the case where noise is introduced into the visual feedback channel (TV monitor), when the communication channel is narrowed and when the task complexity increases.

Pesch, Hill, and Allen (1971) in a series of experiments compared the performance capabilities of underwater manipulator systems with that of a free diver. Performance ratios, in terms of time of task completion, of 1:10 were observed. They attributed these low performance levels to the inability of the human operator in terms of attention and dexterity to control, via joysticks, a multi-degree-of-freedom system, such as a manipulator.

Johnsen (1965) was the first to suggest, explicitly, delegating the task of localized control to a computer. He argued that "Remote manipulators can achieve substantial improvements (in speed and accuracy) through the development of localized control loops which will remove many of the detailed and routine operations from the direct control of the human being." Ferrell and Sheridan (1967) have developed the idea further and coined the term "Supervisory Control." They describe a supervisory controlled system as having three control loops as shown in Figure 2-1.

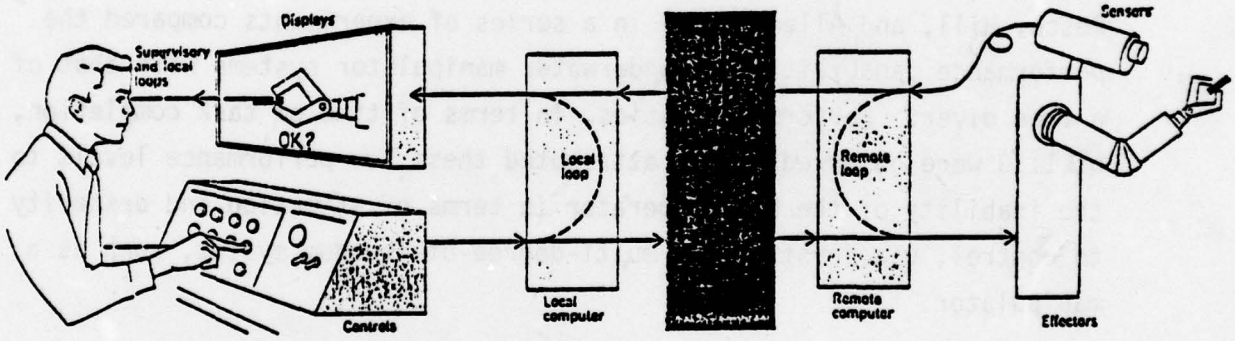


FIGURE 2-1.
SCHEMATIC DIAGRAM OF SUPERVISORY CONTROL
OF REMOTE MANIPULATION

(1) The remote loop, closed through the computer at the distant site, represents the primitive manipulator actions autonomously controlled by the computer.

(2) The supervisory loop, closed through the man, represents his function of intermittently setting goals for the remote device adjusting its characteristics, and, if needed, even guiding the effectors directly. In the other direction the loop carries information about the environment and the activity at the remote site.

(3) The local loop is independent of the remote system, and signifies the operator's use of his computer to model the remote system so that he may embellish the information coming back from the remote site, or predict its behavior and hence improve his supervision. Thus, through mimicry, the local device can provide the operator with "quasi feedback," not subject to transmission delay, noise or bandwith limitations.

The local computer, in addition to serving as an assistant, translates the outgoing and incoming information between a form appropriate for transmission and machine use and a form suited to the man's displays and controls. In this way, we avoid using a more expensive remote computing facility for this purpose.

In the design of such a supervisory system, the problem of the language used for man-machine communication takes a central role; this communication language is the subject of this whole work.

Autonomous Control. The autonomous control approach attempts to produce completely autonomous manipulative systems. Usually, the need for such a system arises in remote space missions, where the time delay in communication would prohibit earth-based control. The work is also directed toward a future, completely automatic, manufacturing process.

The problems for which such a system has to provide solutions are the following: (1) communication of the task, (2) cognition of the environment, (3) problem solving (on an abstract level), (4) planning of solution, (5) execution and monitoring of plan, and (6) procedures for error recovery or problem reporting. All these problem areas have been continuously attacked by Artificial Intelligence researches but until now with only limited success.

The most practical approach attempts to develop higher and higher command languages, ultimately reaching the point where the commands will specify complete tasks. For example, Lozano-Perez and Winston (1977) describe LAMA, a high level command language at MIT. It solves the problem of Object Collision Avoidance by brute force--requiring extensive three-dimensional modeling of all objects in the work area. This system is, thus, an attempt at solving problems 1, 2, 3, and 4 above. Finkel et al (1974) describe AL, a similar system that defines motions between objects by explicit coordinate transformation. AL can be considered only as a step toward problem 1; and in fact, is a more fancy example of "programming approach" to the manipulator control problem. Finally, Ambler et al (1973) describes a similar computer controlled assembly system. This system, however, was unique in that it could recognize the parts it needs from a randomly dumped heap of parts. It would take the heap apart, put the parts in order, and proceed to assemble them by feel, according to a prepared assembly program. It can be instructed to perform a new task with different parts by spending an hour "showing" it the parts and a

day or two programming the assembly manipulations. Amber's system is, thus, a promising attempt at all the six (6) problems previously stated, but at the current state it requires extensive control of the environment in terms of lighting and background and pre-knowledge of all parts in the heap.

Fike and Nilsson's (1970) STRIPS is an example of a symbolic problem solver that attempted to solve robot location problems in a simple environment. It worked well only on simple problems (in terms of the number of operators required in the solution) and Sacerdoti's ABSTRIPS (1975) achieved some success in pushing the complexity barrier by solving the problem in a hierarchy of abstract spaces. Fahlam (1974) describes a problem solving system for block manipulation which demonstrated high solving power for this idealized world of blocks. The key words, however, are idealized and abstract. These problem solvers cannot, at this state, handle the complexity of real world objects in terms of shape, position relations and function.

Table 2-1 contrasts the features for some of the manipulator command languages discussed above. Although these examples and others are promising, the state of the art in Artificial Intelligence is not yet capable to devise a fully autonomous control system for a robot or a manipulator. The problems lie in (1) cognition-recognition of realistic 3-dimensional objects, (2) problem solving in a real and incompletely known environment, (3) planning, and (4) execution with proper recovery from errors, blunders or changes of the environment. Until these problems are solved, the supervisory control mode of manipulators--with the human operator in the loop--is necessary for real time control of novel tasks.

TABLE 2-1

MANIPULATOR COMMAND LANGUAGES

PARADIGM	SUPERVISORY CONTROL	AUTONOMOUS CONTROL	PROGRAMMING SHOP	PROGRAMMING SHOP	MACHINE SHOP
System Name/Source	SMC[1] L/@ PERCEPTRONICS	STRIPS/Fikes Et Al	AL/Finkel Et Al	Alfa/Wang @ GTE Labs	GOERTZ
Applicability					
Simple/Complex Tasks	Complex	Simple	Industrial Assembly Tasks (Medium)	Industrial Assembly Tasks (Medium	Complex
Repeated/General Purpose Tasks	General Purpose	Attempt General Purpose	Repeated Tasks	Repeated	General Purpose
Real/Time/Long Preparation Needed	Real Time Control	Long Preparation	Extensive Programming and Debugging	Long Preparations	Real Time
Needs a Controlled Prearranged Environment?	No	Simple Uncontrolled	Yes	Yes	No
Language/Model					
Task Oriented?	Yes	No	Somewhat	High Level Problem Oriented Language	No Symbolic Language
Hierarchical	Yes	Yes	Yes	Yes	-
Has Sentence Structured Commands?	Yes	No	No	No	-
Has Error Recovery Capability?	Yes	Rudimentary	No	Yes	-
Is Programmable?	Yes	Automatic Problem Solving	Yes	Yes	-
Needs to be Programmed Offline?	No	Problem Statement	Yes - Extensive	Yes - Extensive	-
Programmable in Real-Time?	Yes	No	No	No	-
Has Loops and Variables?	No	Yes	Yes	Yes	-

TABLE 2-1 (CONTINUED)
MANIPULATOR COMMAND LANGUAGES

PARADIGM	SUPERVISORY CONTROL	AUTONOMOUS CONTROL	PROGRAMMING SHOP	PROGRAMMING SHOP	MACHINE SHOP
System Name/Source	SMC^3L/@ PERCEPTRONICS	STRIPS/Fikes Et Al	AL/Finkel Et Al	Alfa @ GTE Labs	GOERTZ
Control					
Planning	By Man	By Machine	By Man	By Man	By Man
Programming	By Man	By Machine	By Man	By Man	-
Monitoring	By Man & Machine	By Machine	By Man	By Man	By Man
Problem Solving	By Man	By Machine Simple Problems	By Man	By Man	By Man
Executing	By Man & Machine	By Machine	By Machine	By Machine	By Man
Input					
Analog Inputs Available?	Joysticks	No	No	No	No
Master/Slave Available?	No	No	No	No	Master Slave
Dedicated/Typewriter Keyboard?	Dedicated	Terminal, Offline	Terminal, Offline	Teletype	No
Feedback					
Immediate Feedback From Input Device	Tactile & Visual	All Feedback Goes to Machine	Tactile	Tactile on Keyboard	Force Feedback
Machine State (Computer)	Visual	-	Visual on CRT	No	-
Manipulator State	Emergency States	-	Visual by Man	No	Visual Through Port
Task and the Environment	Video to Man	-	Visual by Man	Visual by Man	Visual Through Port
Notes	-	*Uses Resolution Method for Problem Solving*			

2.2 Communication Language Model

2.2.1 Requirements of the Model. Before a language design can commence it is our opinion that a clear underlying model has to be adopted. A single overall model gives a straight backbone to the process of language design, system design, dialog formulation and interface design. This is a requirement that can be drawn from the simplicity and consistency requirements emphasized by Falkoff (1973), Kennedy (1975) and Engel and Granada (1975). What are the requirements of the model itself? And, what can we expect to apply to it during the various steps of the design?

The Model we were looking for was not a normative model, (Sheridan and Verplank, 1978) as the man-machine communication process is not sufficiently understood to warrant such a model. It should be a descriptive model of the task, it's perception in the operator's mind, and the communication protocol natural to him.

The model should be decriptive and help understand the following componets and their dynamic interaction:

(1) The task, its components including time and causal relations among them, during task performance.
(2) The human perception of the task prior and during execution (including problem recovery).
(3) The roles played by the human and the assisting computer and how control should be exchanged between time.
(4) The communication protocol, its phases and transitions between them.
(5) The required elements of the abstract language, the structure and relations among these elements, and how they are defined, modified and used.

(6) The required structure of the language implementation.
(7) The required hardware interfaces both from man to machine and the feedback from machine to man.

The modified procedural net model provides insights into all these elements.

2.2.2 Adapted Procedural Nets. The Procedural Nets (PN) described by Sacerdoti (1975) are used, not as a communication model alone, but also are used in the NOAH system as the data structure for a problem solving mechanism and a structured knowledge base for a query answering system that answers questions about the state-of-the-environment and the progress in task execution. In the NOAH system, the PN are used for task description and a machine advisor for an apprentice. In such application, the machine actually knows the task better than the man. Thus, the PN are developed hierarchically, by the machine from the top down to the level of details that the apprentice can perform directly. This development is hierarchical in the sense that it can be developed to differing levels of details for different parts of the same task depending on the level of knowledge of the apprentice about the particular subtask at hand. Sacerdoti's PN provide for splitting and joining nodes to represent parallel subtasks or subtasks for which the order of execution is immaterial. They also provide, at each node a natural language description of what that node is for, i.e., the task to be performed. Finally, they provide ADD and delete lists of assertions about the state-of-the-environment that help simulate the effect of each subtask on an internal representation of the state-of-the-world. These lists are useful for answering questions that the user may ask during the task, they are also useful in the problem solving function which works by resolving conflicts between preconditions and effects of successive subtasks. Most of these complexities are not necessary if we want to use the procedural nets as a communication model

and an internal representation of an ongoing task in a real time command environment.

Figure 2-2 is a static PN representation of a complete task. It is static in the sense that it describes the logical and time relations among the component subtasks, rather than being a snapshot of the activity at some instant. The figure shows the essential ingredients of the procedural nets as they are adapted in this work.

We are using the model to represent on-line, real time activity in a practical task environment, where the operator has the role of a supervisor. To eliminate overloading the operator with several parallel tasks, we eliminated the option of two parallel activities with split and joining nodes. We do not use the net as a base for operator queries about the task and the state-of-the-environment, thus we eliminated the long verbal description of the task associated with each node and also the ADD and DELETE lists that are the basis for the problem solving capability of NOAH.

Nodes in the PN of Figure 2-2 contain only a label by which they can be called and pointers to a "program" describing the sequence of tasks at one lower level. These lower level nodes can in turn be complex nodes themselves; thus, the structure can be extended to any level of complexity.

The same structure can be used to derive a communication model for man-machine communication as shown in Figure 2-3. The figure depicts one possible role assignment between man and machine. The human operator decides on the task at the top level in the hierarchy, he then develops it in his mind down to the level of detail that the machine (computer) can accept. He communicates this intermediate level plan, which is shown in the figure as a series of nodes linked together by time procedures, to the computer.

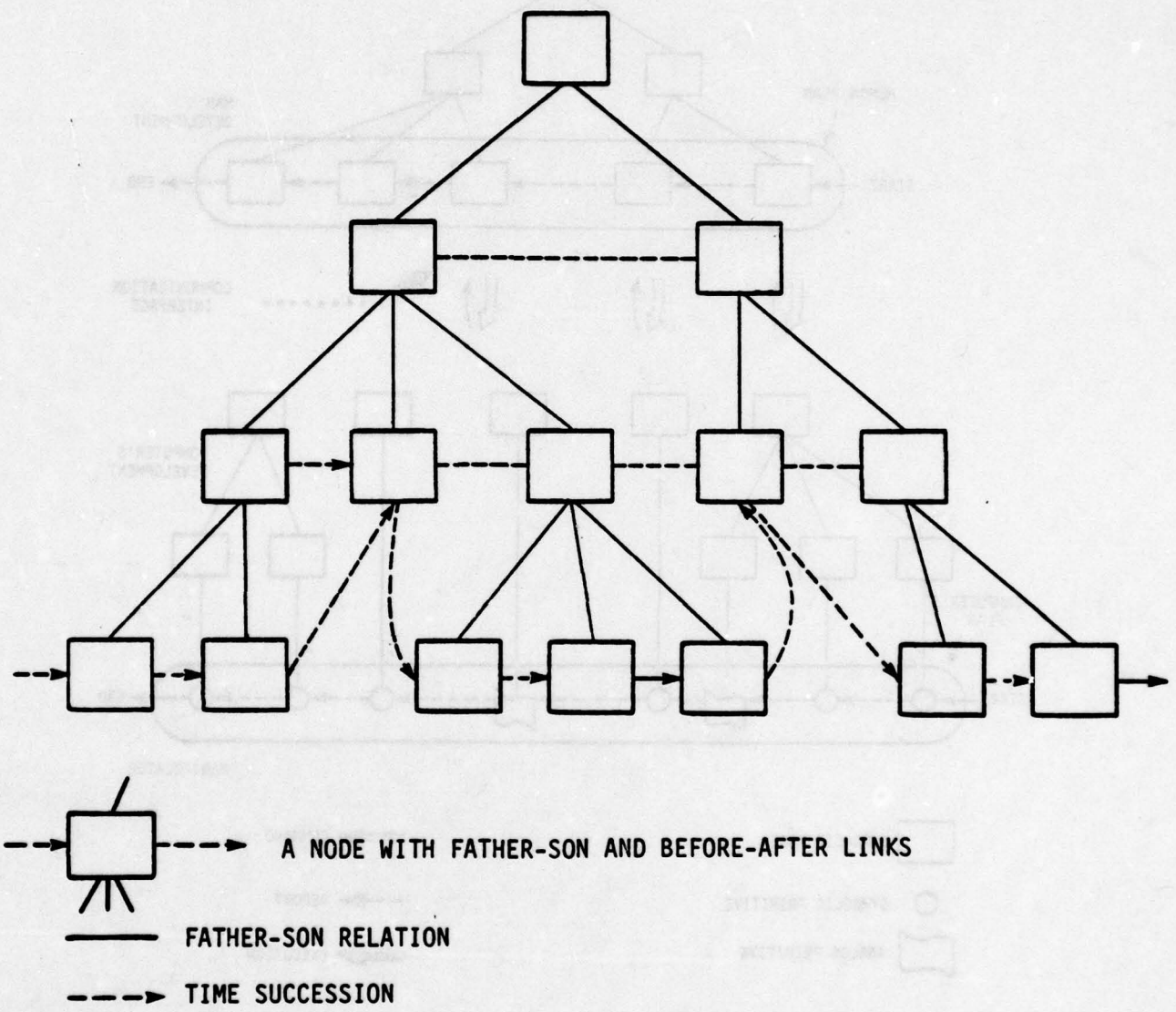


FIGURE 2-2.
A SIMPLIFIED PROCEDURAL NET

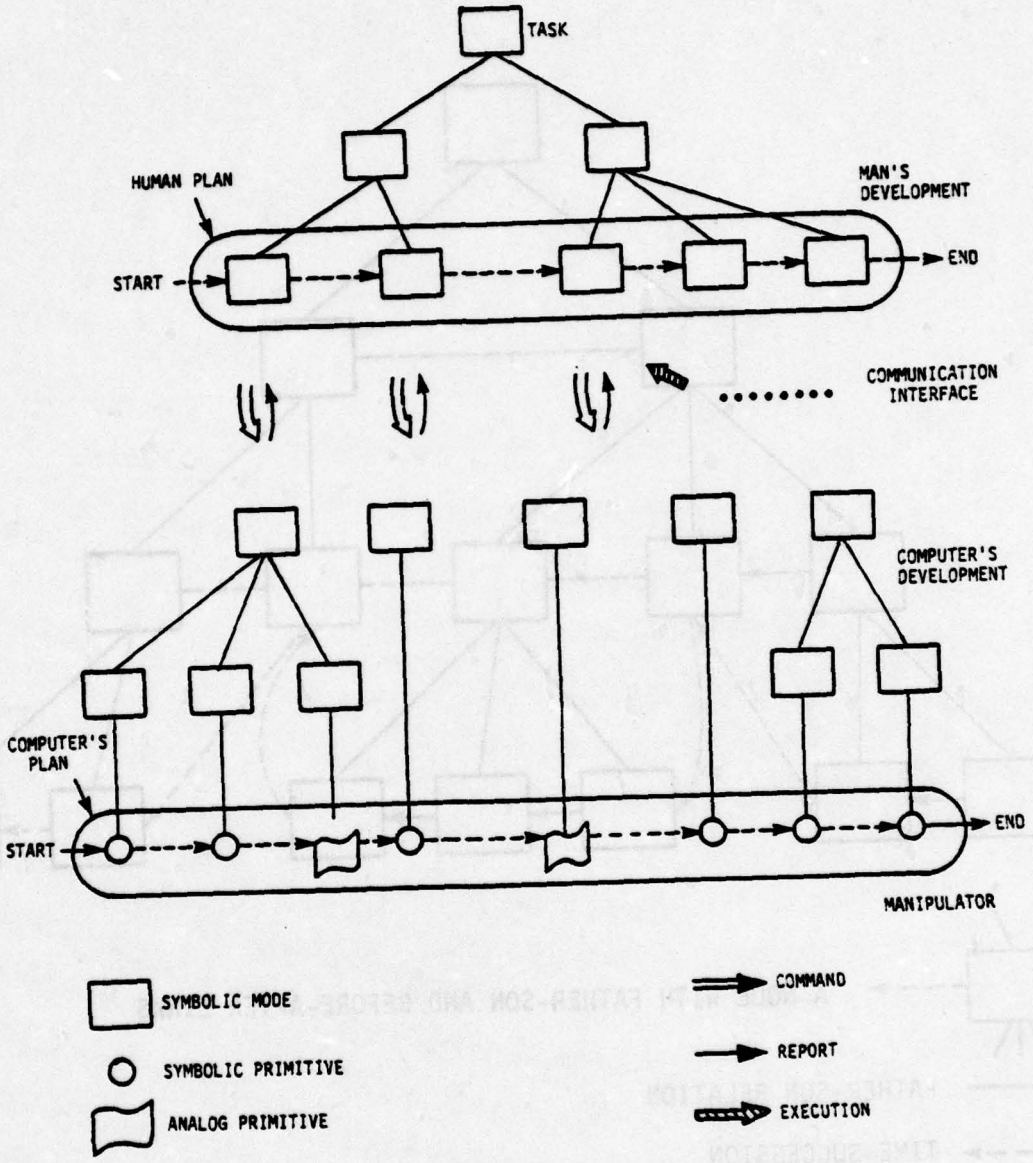


FIGURE 2-3.
THE SIMPLIFIED MAN-MACHINE COMMUNICATION MODE

The computer, having previously stored the description of each subtask at that level, develops the plan to the level of primative actions, which can be communicated directly to the manipulator. (We are interested in the man-machine communication aspects, thus we will ignore the distinction between the computer and the manipulator itself; they will be considered one system.)

The level at which the man-machine interface occurs determines what was termed the communication mode (Verplank, 1967). Figure 2-4 shows various possible placements of the man-machine interface adapted from Sheridan and Verplank (1978) which is one of the main language design decisions. If it is placed at a very low level (see Figure 2-4A), where the operator controls joysticks which cause link movements, then we have "direct control" and a computer is actually unnecessary. If the communication is done at a very high level, shown in Figure 2-4B, then we have "symbolic control" or considering the fact that in this case a computer does a large part of the plan development and monitoring, this is sometimes called "automatic control." In controlling a manipulator, either control mode alone, cannot be sufficient. As Verplank (1967) has shown, both analogic ("direct") and symbolic commands are necessary for effective communication. Some tasks--those completely prespecified--can be better called by a symbolic command and others--those involving complex geometrical motions that are not repeated--can be better specified by direct control. This is shown in Figure 2-4C, and in the different types of tip nodes in Figure 2-3. Figure 2-4D presents an interesting twist on the man-machine control allocation issue. Here the man sets up the goal of the task, at the top level, the computer plots the strategy and the actual execution is again given to the man. This control allocation is the one adopted in NOAH where the computer knows how to perform a set of tasks, the human apprentice sets the goal, and then requests from the computer the detailed plan (sequence of actions) that will accomplish the goal. He then goes ahead and executes the actions

himself. This discussion shows the flexibility provided by the modified PN model in describing control allocation strategies.

The real time, on-line requirement and the varied, changing environment of undersea manipulation task disallow extensive preprogramming of complete tasks or subtasks. Thus, we consider the top down, "programming shop paradigm" of specifying tasks for a manipulator to be inapplicable. The order of development must be from the bottom up, as the user of the system has first to define a task before he can call it into use. He can then define gradually higher level tasks that use the simpler tasks as steps. To facilitate such a process the man-machine language has to provide the following components:

(1) A set of analog and symbolic primitives.
(2) A way to define or specify variable primitives (a point in space).
(3) A way to modify and correct previously specified primitives.
(4) A way to hierarchically define chains of primitives or chains.
(5) An editing capability for chains.
(6) A control mechanism for execution of a complex command.
(7) A feedback process to report progress problems and completion.

All these components are needed to implement a communication process based on the PN model shown in Figure 2-2.

2.3 Command Language Analysis and Design

2.3.1 Overview. The language is intended to close the gap between the high level task oriented terms in which the human operator conceives a manipulative task, and the low level link-motion primitive commands that the manipulator can execute directly. The focus of this third year's

effort has been on the analysis and comparison of the command language features required for effective control of a semi-autonomous, computer-aided manipulator. The effort also included the implementation and software support.

2.3.2 Communication Function Analysis. The way in which the operator interacts with the manipulator affects operator performance requirements. Basically, the functions of the operator in supervisory role are:

(1) Plan the task.
(2) Communicate (command).
(3) Monitor.
(4) Intervene when appropriate.

Several approaches for task classification and requirement specification in performing above functions have been considered to identify communication requirements.

Typically, an underwater mission may include one of the following tasks (Pesch, 1972; Bien and McDonough, 1971):

(1) Installation and maintenance.
(2) Recovery and retrieval.
(3) Salvage.
(4) Surveillance.
(5) Habitat development.

Breakdown of the tasks into sub-task level, according to the type of manipulator motions, forces, the modes of feedback and the tools needed was performed by Schneider (1977). Given Schneider's results it is evident that vision contributes to every task, that gross force is next most important, that special tools are needed for about half the tasks.

A general next-level down task analysis has lead to the following command activities necessary to perform a designated mission:

(1) Position the effector.
(2) Transfer a load.
(3) Track a moving target.
(4) Apply force to, or position the manipulator.
(5) Configure, orient, and position the manipulator.
(6) Configure, orient, and position the end effector.
(7) Operate the end effector to grasp, hold, turn, carry, align or perform special operations.
(8) Remove/replace/adapt the end effector.
(9) Detect and avoid obstacles in the path of motion.
(10) Coordinate multi-manipulator or multi-effector machine.

To coordinate all these activities, the communication language design requires collaboration with different control modes, levels of feedback and the extent of man-machine interaction in controlling the manipulator system.

Our work on a theoretical model for man-machine communication lead to the development of a Shared Man-Computer Control and Communication Language (SMC^3L) and a control system. The system integrates several concepts from computer science, psychology and human factors theory into a coherent communication aiding system. Hierarchical decomposition of tasks, subtask naming, sentence structure, mixed initiative and hierarchical feedback are used to facilitate easy, speedy, and natural man-manipulator communication.

2.3.3 Approach for Language Design. The specific problem this research tried to solve is the development of a model and the necessary mechanisms for a communication language between man and machine in the supervisory mode of the manipulatory control paradigm.

The Procedural Net Model. The model used to describe the role of a man-machine command language in a supervisory controlled manipulator is an adaptation of the concept of procedural net developed by Sacerdoti (1973) and 1975) to represent plans of manipulative actions. Using this representation, a problem domain is described as a hierarchy of procedural nets, each representing some task in the problem domain in terms of its goal, its component subtasks, and their relation to the environment.

The procedural net model suggests a basic structure for the language. It is a task oriented language with provisions to define tasks as a sequence of subtasks. Together with a set of primitive actions defined directly as manipulator motions, such a recursive organization provides the operator with capability to define tasks at arbitrary levels of complexity. The operator is doing the high-level planning and the computer handles the details.

Therefore, the procedural nets used in the present model are a simplified version of the model suggested by Sacerdoti. The simplifications were made to provide affluent man-machine communication. This is in contrast with NOAH in which the procedural nets are used as a method to represent the tasks inside the computer. Additionally, the present application is oriented toward real time generation of commands on line by an operator where simplicity is paramount, while in NOAH the procedural nets are constructed and debugged offline at leisure. The simplification includes: elimination of parallel activities, no explicit goal statement on the nodes and no logical conditions that determine what actions will be executed.

Language Constructs. The specific constructs of primitives and variables that have to be incorporated in a command language are task specific and have to be determined by task analysis and experimentation. For the underwater manipulation application we have utilized previous analysis done by

Pesch et al (1970), Bin and McDonough (1971), Ocean Systems, Inc. (1977) and Schneider (1977), and came up with the set of primitives described below.

According to Verplank (1967) Hill (1971) and others it is necessary to have both symbolic and analog commands. Symbolic for discrete actions that are either very small or are completely prespecified, and analog for commanding manipulative actions. Within the analog primitives there are also two classes of the control assignment: Direct control and "spatial control". In "direct control" the operator controls with his analog input devices (two 3-degrees-of-freedom joysticks) the angles of each link of the manipulator. In "spatial control" he moves the end effector - the gripper in the cartecian three-space x y z. Experiments under this contract, performed in previous years (Berson et al, 1977), have shown than for some tasks "direct control" is preferable and for others the "spatial control" mode gives better performance.

The symbolic primitives come in complementary pairs GRASP, RELEASE close and open the gripper, FORWARD, BACKWARD, move the gripper forward or backward along its axis a fixed distance. In addition to these discrete primitives the language contains a class of variable primitives. These are POINTS. A POINT is a labeled point in link space, i.e., a specified manipulator configuration. Such a POINT can be defined - a label given to a particular manipulator configuration, it can be modified and it can be used. When a POINT is called the manipulator moves back to that point in space. The labels have to be short, simple and a few in number to avoid loading user's memory, thereby causing errors and delays. We have selected simple digit numerals as labels for points and other variables. Previous experiments at Perceptronics (Berson, et al, 1977) have shown that if it takes several bottom pushes to specify a name, the user would prefer to perform the task by analog command instead just to save time.

The main contribution of the model is suggesting how to construct composite task oriented constructs to be build hierarchically from simpler ones. The complete task at the interface between man and machine is represented as a sequence of subtasks ordered in time. Thus the man/machine communication protocol would be essentially such a sequence; the operator commanding the computer to perform the particular sequence by giving the commands in order. Each command is a conceptually complete task (Foley and Wallace, 1975) and is either a primitive or a construct - which we termed "chains" and "paths".

There is an important issue that must be emphasized at this point. It is the need to predefine the concepts before use and thus a bottom up order of concept development. When a person communicates with another person in natural language saying, e.g., "Bring over that chain", the listener uses much previous knowledge to interpret the command. He uses syntactic and semantic knowledge to understand the action "bring" and object "chair" in the visual field and coordinate his movement in 3D space toward the chair, how to pick it up and how to maneuver it in the room, and finally, he uses social knowledge to control the overall speed of motions and the exact final placing of the chair. Current manipulator systems lack most of these skills and thus every task must be specified in detail before it can be used. Further, in the real-time, on-line novel environment, as it exists for underwater tasks, very little preplanning can be done. Sizes, orientation and positions are not known before the dive and thus preparation of complete tasks are impractical. This is unlike the industrial environment where the environment can be controlled enough to allow complete programability (Nevin and Whitney, 1978). Suggesting top-down definition on site would also be impractical. The user would prefer to start the task immediately using direct control or low level primitives (Berson et al. 1977).

We conclude that a bottom up order of task definition is the natural sequence which the model suggests. Furthermore, a definition capability of new tasks should be smoothly integrated in the language and communication protocol so that the user can use available subtasks or define new ones easily when he identifies an opportunity to save time.

There are two composite constructs in the language. A "chain" is a predefined and labeled sequence of primitives and/or chains. A "Path" is a predefined and labeled sequence of analog motions that are stored in memory. These two constructs are duals of each other in that one combines symbolic substructures (a path can be a substructure of a chain) while the path stores analog motions for later reply.

According to the previous discussion, these constructs require the following facilities in the protocol:

(A) A definition capability.
(B) A labeling capability.
(C) A display for review capability.
(D) An editing capability - for error correction.
(E) A calling procedure.
(F) A progress display capability.

A through E are self explanatory. The requirement F is necessary to keep the operator in evident control (Kennedy, 1975). If a complex process is controlled automatically by the computer the user must be kept informed about the progress of the task and what goes on at the moment. In this system, we display the complete chain currently active; further research is needed, however, to determine the optimal amount of such feedback.

2.3.4 General Language Design Steps. The discussion in the previous sections can be summarized in a procedure that specifies the sequence of tasks that have to be done when designing a computer assisted man/machine command language and structure.

(1) Task analysis.
    1.1 Identify a complete set of system tasks.
    1.2 Identify repeated subtasks.
    1.3 Separate analog from discrete subtasks.

(2) Analog primitives determination.
    2.1 Determine the number of analog inputs needed concurrently.
    2.2 Determine the need for several analog modes.
    2.3 Determine the analog input devices and their assignment (sliding potentiometers, joysticks, and master/slave, etc.)

(3) Symbolic primitives determination.
    3.1 Define discrete primitives (with few or no parameters) to correspond to repeated subtasks in 1.3.
    3.2 Define variable discrete primitives.

(4) Determine composit constructs.
    4.1 Determine analog composit constructs (paths) with facilities to define, label, modify, and display them.
    4.2 Determine symbolic composite constructs (chains) with corresponding facilities to define, label, modify and display them.

(5) Define command structure to be "sentence structured" with simplicity and consistency through all commands. Minimize the number of verbs and nouns.

(6) Design an integrated analog and symbolic interface to correspond physically to the sentence structure and provide immediate feedback and conceptual continuity.

(7) Design the interaction protocol, exchanging control between man and machine in mixed initiative control allocation. Allow for queueing of input commands and natural transfer from editing and modification to execution and back.

(8) Define feedback format which are consistent and adhere to the priniciple of visual continuity (Foley and Wallace, 1975) for the following information:

8.1 System state.
8.2 Command entered.
8.3 Error messages.
8.4 Cues and event reports.
8.5 Sensor information.
8.6 Chain and path displays.
8.7 Queue display.

The next section describes the details of the language for the control of a manipulator, while Appendices A and B describe the details of the hardware and interface formats.

2.3.5 Command Language Design: SMC^3L. The command language consists of two major elements: symbolic and analogic commands. In the first class, symbolic commands are executed by discrete pushbotton actions, resulting in discrete and defined arm motions. In the second class, analogic commands are executed by continuous movement of joysticks, resulting in analogous manipulator movements. However, the language also includes the

capability to "define" manipulator configurations, called "points", or trajectories, called "paths." This "define" function permits the operator to first move the arm using the analog joysticks, and then to repeat the same motion, or reverse the motion by using symbolic commands.

The symbolic commands consist of two categories of commands: Primitive and Variable. The primitive commands include the conceptual unitary motions. For the tasks used in the present experiment, the following primitive commands were used:

FORWARD
BACKWARD
GRASP
RELEASE
ROTATE CLOCKWISE
ROTATE COUNTERCLOCKWISE
MANUAL

The Variable category of commands are the user-defined commands which provide the capability to construct task specific commands. When employing a variable command, the operator conceives of the task at a high level of abstraction and the computer implements the lower level details of the pre-recorded manipulations. These Variable commands include:

POINT - moves the manipulator to a previously recorded position.
PATH - moves the manipulator along a previously recorded trajectory.
REVERSE PATH - moves the manipulator along a path from and to beginning.

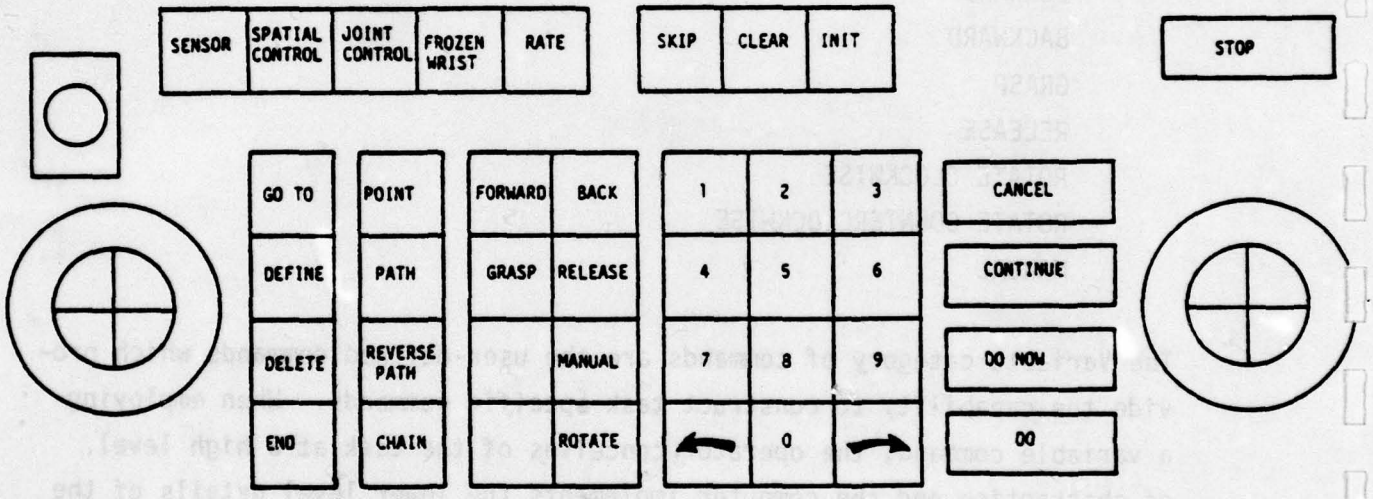


FIGURE 2-5
OPERATOR KEYBOARD

This chain would cause the manipulator to move to point 6, which has previously been recorded in front of the valve, and to open the value one complete revolution (360^{o}).

Given the two categories of symbolic commands, the command language was designed to have a general "verb-noun-parameter-terminator" syntax. The syntac was preserved and facilitated by the keyboard as shown in Figure 2-5. All commands presented here specify conceptually closed tasks. This is true when activating primitives, composite tasks or when changing the system state. Each command is given as an independent sentence, not unlike an imperative natural language sentence. This syntax follows Foley and Wallace (1975) and Treu (1975) recommendations.

In addition to the pre-recorded constructs "chains" and "paths", the system may contain a one time dynamic ordering capability - a queue. The supervisory mode of control discussed here results in that large parts of the task are controlled automatically by the computer at its own pace. A queueing capability allows the operator to enter the symbolic commands at his best pace, even running ahead of the execution pace, and the computer queues the entered commands and executes them in a simple FIFO order.

Adding the queue feature to a system adds other servicing requirements that are the results of the principle "every error can be undone". The following features have been incorporated in our system:

(A) Clear the queue.
(B) Skip one command in the queue.
(C) Insert a command at the head of the queue (that is the DO-NOW terminator).

3. EXPERIMENTAL STUDIES

3.1 Introduction

3.1.1 Objective. The overall objective of this program is to evaluate the man-machine communication language developed for computer-aided remote manipulation. The emphasis of the current year's work was twofold. First, the command language was evaluated in a work situation that simulated the type and quality of environmental feedback typically available to the operator. Secondly, the additional machine-state feedback required to use the high level, symbolic commands was experimentally evaluated. The implementation and results of these efforts are described in this chapter.

3.1.2 Command Chaining. The "chain-in-chain" feature expands the SMC[3] language construct to allow higher levels of the task hierarchy. Basically, chains are sequences of specific primitive commands which are grouped and defined by the operator as repeatable subtasks. They provide the operator with the capability to define a useful sequence only once and later use it repeatedly by recalling it as a unit. The computer then performs that stored sequence automatically, possibly faster and with less probability of errors. The "chain-in-chain" capability allows chains to call chains at any level. This feature permits the operator to construct relatively complex tasks by combining a sequence of chains from usually simpler subtasks.

3.1.3 Machine State Feedback. The symbolic command language alters the cognitive and task requirements of the manipulator operator. Most of the new requirements involve communicating with the computer. Specifically, in supervisory control, the operator not only provides direct analog control of the manipulator's movements, but also (1) selects and confirms the

available computer-assisted functions, (2) determines what control mode is currently operating, and (3) monitors the progress of automated routines, etc. The purpose of the present effort was to establish the amount of machine-state feedback required for effective man-machine communication when using the high level, chained commands.

Currently, the feedback for chain commands is provided through a CRT display of the list of commands that make up the chain. It includes both the full delineation of individual commands and their sequence, and display of the current operational command (Figure 3-1).

3.1.4 Environmental Feedback. In previous work of the current program, the command language evaluation was conducted in a situation where the operator had a direct and unobscured view of the manipulator and the work space. Direct viewing conditions provided the operator with a wide variety of visual and auditory feedback that is unavailable in situations where remote manipulators are typically employed. An indirect viewing situation (TV monitor), especially in a visually occluded environment, offers a more appropriate command language test environment and raises important issues for computer aiding in remote manipulation. For example, research has shown that the degradation of environmental feedback associated with indirect viewing impairs performance with manually controlled manipulators (Hill and Sword, 1973). However, the effect of indirect viewing on performance with automatic commands remains unknown. Well designed, computer-aided functions should mitigate the deleterious effects of indirect viewing by allowing less dependence on high quality, environmental feedback. The present command language was designed with this goal in mind, and its evaluation under indirect viewing was a central portion of the current year's work.

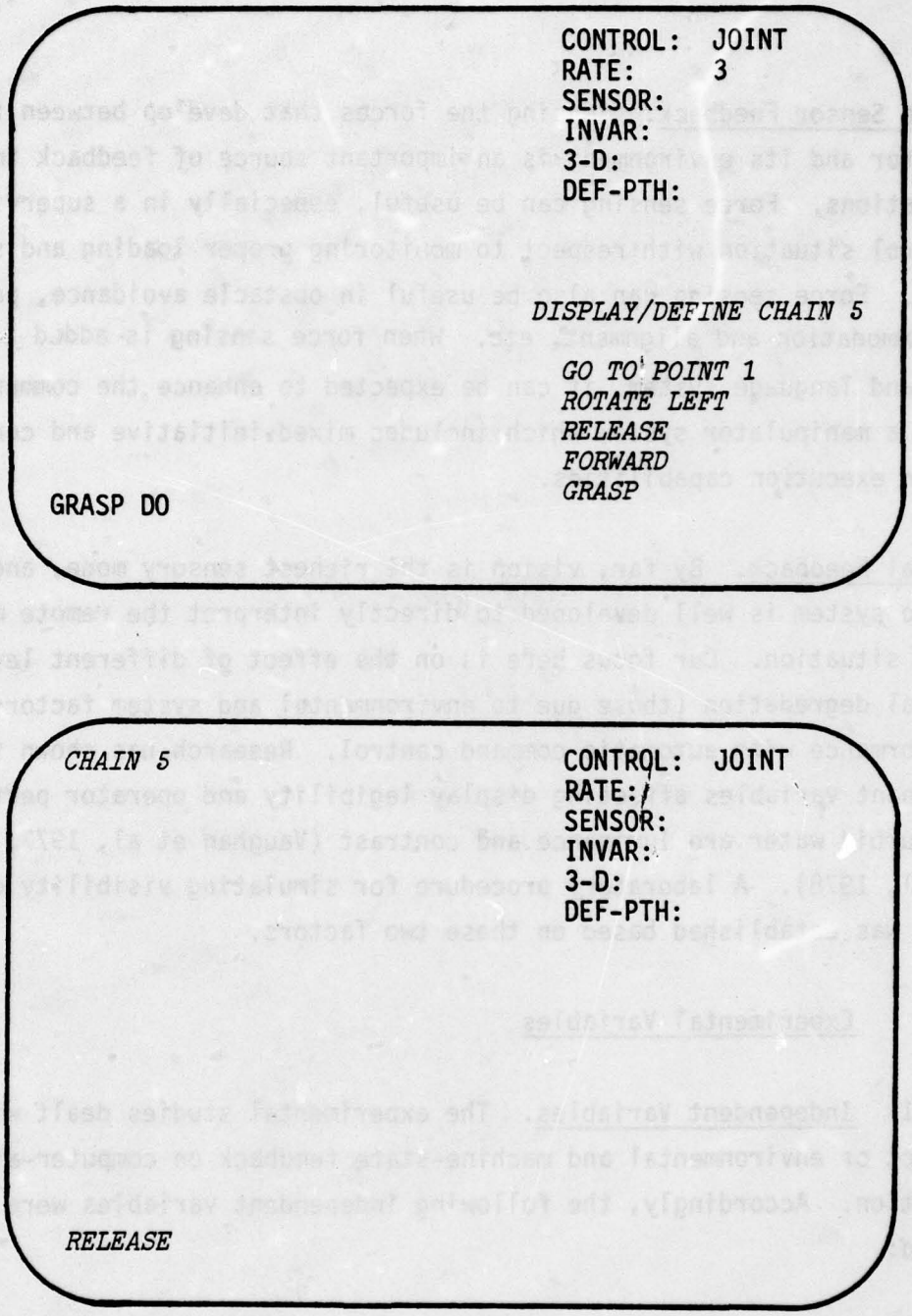


FIGURE 3-1.
MACHINE-STATE FEEDBACK DISPLAY DURING CHAIN DEFINITION (TOP) AND DURING CHAIN EXECUTION (BOTTOM)

Force Sensor Feedback. Sensing the forces that develop between the manipulator and its environment is an important source of feedback in many situations. Force sensing can be useful, especially in a supervisory control situation with respect to monitoring proper loading and safe contact. Force sensing can also be useful in obstacle avoidance, position accommodation and alignment, etc. When force sensing is added to the command language system, it can be expected to enhance the communication with a manipulator system which includes mixed-initiative and conditional-chain execution capabilities.

Visual Feedback. By far, vision is the richest sensory mode, and the video system is well developed to directly interpret the remote manipulation situation. Our focus here is on the effect of different levels of visual degredation (those due to environmental and system factors) on performance with automatic command control. Research has shown that the dominant variables affecting display legibility and operator performance in turbid water are luminance and contrast (Vaughan et al, 1977; Pepper, et al, 1978). A laboratory procedure for simulating visibility degredation was established based on these two factors.

3.2 Experimental Variables

3.2.1 Independent Variables. The experimental studies dealt with the impact of environmental and machine-state feedback on computer-aided manipulation. Accordingly, the following independent variables were investigated:

(1) Command Mode

(a) Manual (Fixed) - allowing basic primitive commands with direct joint control or resolve motion rate control.

(b) Automatic (Variable) - allowing both analogic and symbolic commands within single level chaining.
(c) Automatic - allowing both analogic and symbolic commands with full command language features.

(2) Machine-State Feedback

(a) "Minimal State Feedback" - display of entered command in definition period and during execution.
(b) "Full State Feedback" - display of queue list of commands as well as the entered command in definition period and display of current operational command in execution.

(3) Visual Feedback

(a) Normal viewing - high illumination and contrast.
(b) Degraded viewing - low illumination and contrast.

These variables were evaluated in a sequence of three-phase experiments. The first phase served as a practice evaluation under indirect viewing conditions. The second phase was the evaluation of machine-state feedback requirements. The third phase was the evaluation of both command structure and visual viewing conditions. The design and results for each experiment will be discussed in the following sections.

3.2.2 Performance Measures. Performance measures consisted of task completion times and errors (type and amount) committed by the participants in performing assigned tasks. The times expended in the constituent elements of task operation were recorded and the following error types were observed:

(1) Improper contact.
(2) Dropping the object.
(3) Inappropriate command entry.
(4) Improper sequence of commands.
(5) Go to wrong point, path, or chain.

To supplement the manual data collection, an automated data collection program was developed to record the following event sequences:

(1) Time and type of commands typed at the subject's keyboard.
(2) Time and type of commands executed by the system.
(3) Beginning and end of time intervals when joysticks were operated.
(4) Beginning and end of time intervals when the manipulator was in motion.

These data items provide measures of the following information dimensions:

(1) Functional Allocation - This is the partitioning of the relative control contributions made by man and by machine.
(2) Total Travel Time - This measures the portion of task completion time during which the manipulator is moving.
(3) Type and Time of Keyboard Entries - This allows a determination of the frequency of use of each command type. Also, entry errors are indicated by cancelled commands.

3.3 Experiment 1: Indirect Viewing, Ring Placement Training Tank

The second year's experimental results indicated that automatic command control can improve task performance for a number of remote manipulator tasks in a direct viewing condition. As a baseline study for indirect

viewing, a 2-view, black and white TV viewing system was used, based on its superior performance rank among various viewing systems for remote manipulation (Freedman, Crooks and Coan, 1977). To provide the capability for indirect viewing, the operator's control console was placed in a room adjacent to the manipulator task room. As shown in Figure 3-2, two 12" (diagonal) video monitors are placed above and in front of the control panel. Two video cameras are placed approximately at a 90° orientation in the work space to provide two distinct views of the work space on the video monitors. No camera control (pan, tilt, or zoom) is provided, which prevents any changes in the camera views. A 9" video monitor is placed below the two 12" monitors. This smaller monitor is used to display the alphanumeric feedback concerning communication with the computer control programs.

3.3.1 Practice. All four subjects who participated in this experiment had participated in the previous experiment under direct viewing situations, and had learned the basic skills for controlling the manipulator and using automatic commands. During the practice period, the subjects practiced basic manipulation tasks in a 2-view console. Total practice time was conducted in sessions of approximately 8 hours for each subject.

3.3.2 Task and Procedure. During the experiment, the subjects used the various command modes to perform a ring placement task which required subjects to pick up metal rings (2" hole diameter) and place them on horizontal and vertical post (.75" diameter), as shown in Figure 3-3. The ring placement task was used because it involved a variety of basic manipulation elements (i.e., gross movement, fine alignment, grasping and carrying, etc.). Also, the repetitive nature of this task provided the opportunity for the participants to practice and capitalize on the advantages of the symbolic commands with such a task.

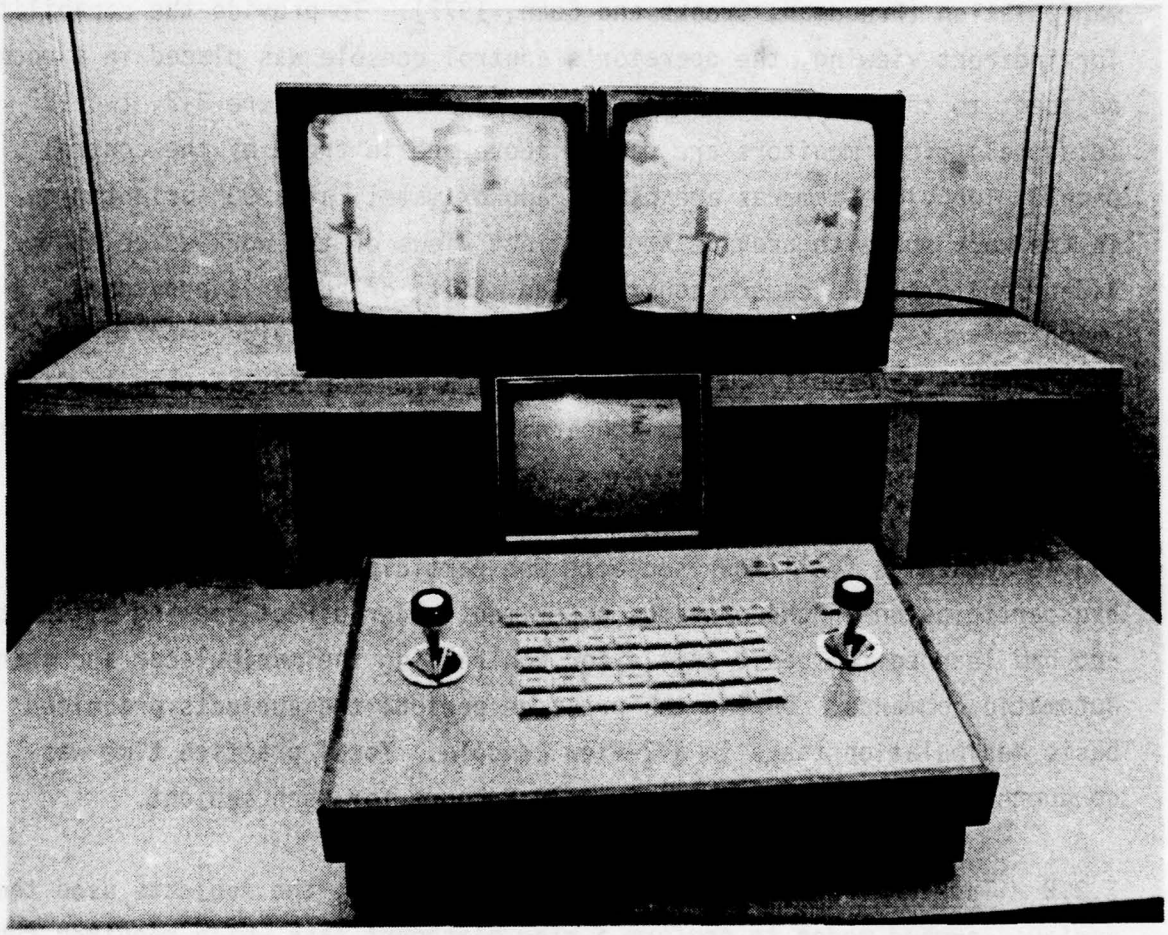


FIGURE 3-2.
OPERATOR CONTROL STATION
FOR INDIRECT VIEWING

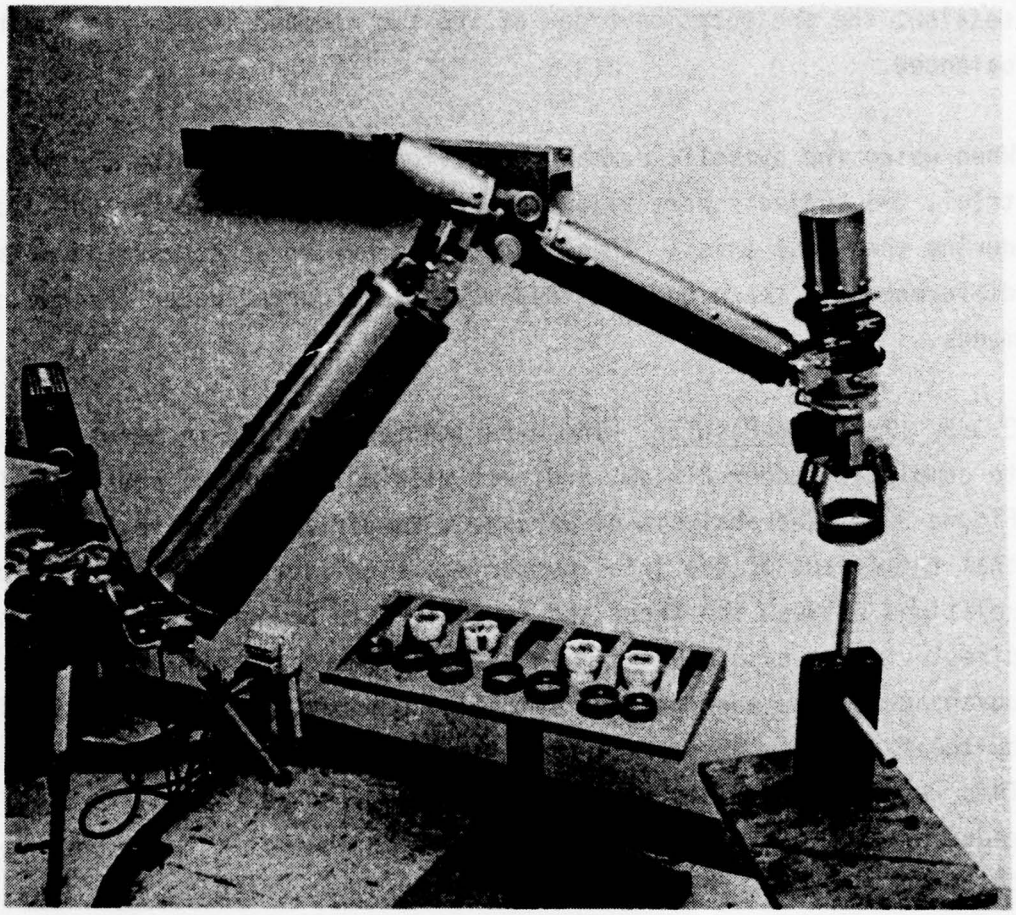


FIGURE 3-3
INDIRECT VIEWING RING PLACEMENT TASK

A 2 x 3 x 2, repeated measures design was used to examine two levels of the command over three sessions, each for two trials. Within each session, the presentation order of the two command levels was counter-balanced.

When using the symbolic command condition while performing the second trial, the subjects were allowed to use any points or paths defined during the first trial. This procedure provided an assessment of any differences in learning rate that may have occurred under the two command modes.

3.3.3 Training Results. The experimental results, in terms of mean time to completion under 2-view, indirect viewing condition, are shown in Figure 3-4. The analysis of variance, summarized in Table 3-1, showed that the effect of the trial number was significant ($P < 0.01$). The time relations between the first and second trials demonstrated (as in the direct viewing case discussed in the last year's report) the definite advantage of the user-defined symbolic commands. This is fully explained by the fact that there was a significant mode-by-trial (AXC) interaction. This result illustrates that automatic commands are less effective in reducing task time in first trials in which automatic functions have yet to be defined, but that they are most useful in reducing task time in later trials where automatic functions are well defined and utilized. However, the second-trial percentage improvements are smaller than in the direct viewing experiment of the previous year. This may be due to the fewer (3) repetitions involved in current ring placement task than those (6) in the previous year.

The cost of defining automatic functions including points and paths was found to increase the initial trial task times by an average of 22%. Although this time cost of defining the commands, when compared across

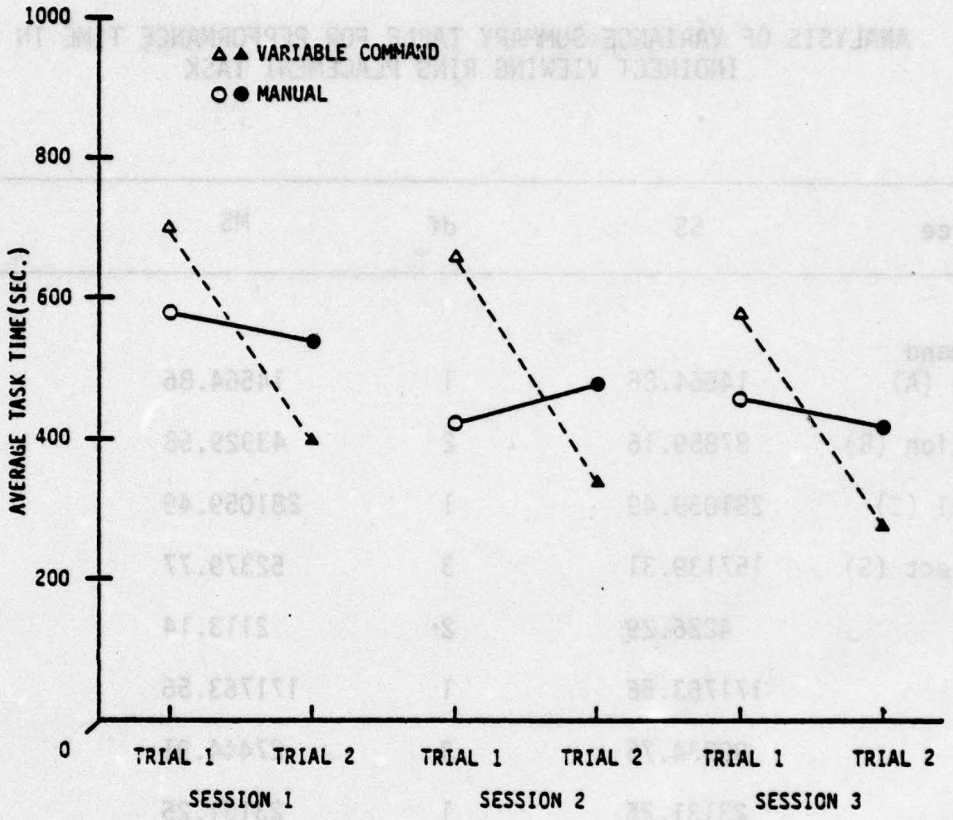


FIGURE 3-4.
PERFORMANCE TIME OF INDIRECT VIEWING RING PLACEMENT TASK AS A FUNCTION OF COMMAND MODE AND SESSION.

TABLE 3-1

ANALYSIS OF VARIANCE SUMMARY TABLE FOR PERFORMANCE TIME IN INDIRECT VIEWING RING PLACEMENT TASK

Source	SS	df	MS	F
Command Mode (A)	14564.86	1	14564.86	< 1
Session (B)	87859.16	2	43929.58	3.67
Trial (C)	281059.49	1	281059.49	62.8*
Subject (S)	157139.31	3	52379.77	1.91
AxB	4226.29	2	2113.14	< 1
AxC	171763.56	1	171763.56	20.65**
AxS	82334.75	3	27444.91	
BxC	23131.25	1	23131.25	2.36
BxS	35870.05	3	11956.68	
CxS	13426.97	3	4475.66	
AxBxC	12231.03	2	6115.52	< 1
AxBxS	158220.53	6	26370.09	
AxCxS	24955.14	3	8318.38	
BxCxS	58932.54	6	9822.09	
AxBxCxS	90729.63	6	15121.60	

* $P<0.01$

** $P<0.05$

TABLE 3-1

ANALYSIS OF VARIANCE SUMMARY TABLE FOR PERFORMANCE TIME IN INDIRECT VIEWING RING PLACEMENT TASK

Source	SS	df	MS	F
Command Mode (A)	14564.86	1	14564.86	< 1
Session (B)	87859.16	2	43929.58	3.67
Trial (C)	281059.49	1	281059.49	62.8*
Subject (S)	157139.31	3	52379.77	1.91
AxB	4226.29	2	2113.14	< 1
AxC	171763.56	1	171763.56	20.65**
AxS	82334.75	3	27444.91	
BxC	23131.25	1	23131.25	2.36
BxS	35870.05	3	11956.68	
CxS	13426.97	3	4475.66	
AxBxC	12231.03	2	6115.52	< 1
AxBxS	158220.53	6	26370.09	
AxCxS	24955.14	3	8318.38	
BxCxS	58932.54	6	9822.09	
AxBxCxS	90729.63	6	15121.60	

* P<0.01

** P<0.05

sessions, tends to decrease with practice, a larger sample size would be needed to demonstrate statistical significance of this trend. Results from subject interviews suggest that practice tends to facilitate those cognitive processes essential to computer-assisted operation to a greater degree than those motoric processes to which manual operation is largely confined, and that facilitation from computer assistance will be most effective for execution of those tasks requiring a high degree of repetition.

In addition to recording performance time, contact errors committed during the performance of the ring placement tasks were also recorded. The number of contact errors for each task under indirect viewing is shown in Figure 3-5. Although there is a similar trend of this measure within trials compared to that of the task time, no effect reached significance based on the analysis of variance upon limited data. This non-significant result is a representation of effects that get buried in the varied actions that the operator does during a simple trial (in which errors are rather infrequent events) and should not necessarily lead to the conclusion that automatic commands have not been as effective in reducing errors as in reducing task time for the ring placement task.

3.4 Experiment 2: Machine State Feedback

The purpose of this second experiment was to determine the effectiveness of providing the operator with CRT-displayed information of machine states. In the current design, automated functions and status are summarized at a conceptual level similar to the commands used. The "minimal feedback" condition identifies only the specific subtask under current execution, as well as input verification. The "full feedback" condition also includes the feedback display of all previously-defined chain components during both definition and execution period (Figure 3-1).

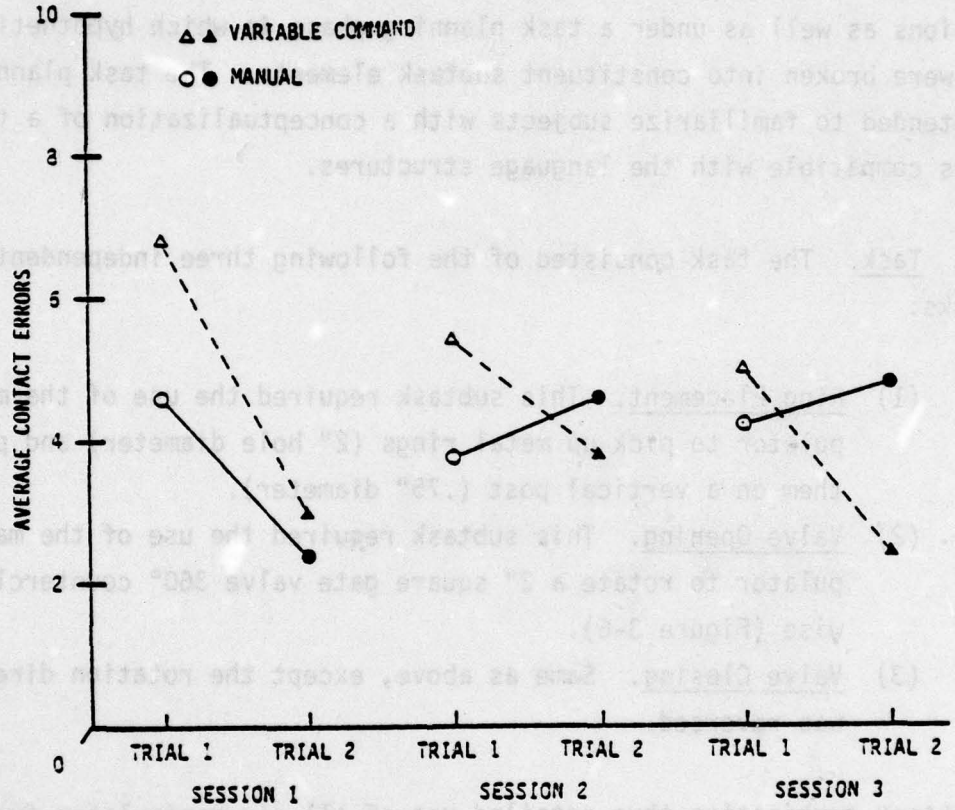


FIGURE 3-5.
CONTACT ERRORS OF INDIRECT VIEWING RING PLACEMENT TASK AS A FUNCTION OF COMMAND MODE AND SESSION.

3.4.1 Practice. Eight subjects participated in the experiments. All the subjects had practiced the ring placement task under indirect viewing conditions as well as under a task planning phase in which hypothetical tasks were broken into constituent subtask elements. The task planning was intended to familiarize subjects with a conceptualization of a task that is compatible with the language structures.

3.4.2 Task. The task consisted of the following three independent subtasks:

(1) Ring Placement. This subtask required the use of the manipulator to pick up metal rings (2" hole diameter) and place them on a vertical post (.75" diameter).
(2) Valve Opening. This subtask required the use of the manipulator to rotate a 2" square gate valve 360° counterclockwise (Figure 3-6).
(3) Valve Closing. Same as above, except the rotation direction was reversed.

The subtask combination thus entailed use of all six manipulator functions, demanding the utilization of a wide range of manual skills and encompassing several qualitatively distinct operations, demanding the utilization of cognitive skills (e.g., to differentiate between gross motion and fine movements, to define and execute subtask elements and to control with mixed initiative, etc.).

3.4.3 Procedure. Subjects performed the experimental task under both "full feedback" and "minimal feedback" conditions within each of the two sessions on consecutive days. The order of feedback conditions was counterbalanced both within and across subjects. (For example, a subject operating first with "full feedback" on Session 1 would begin with

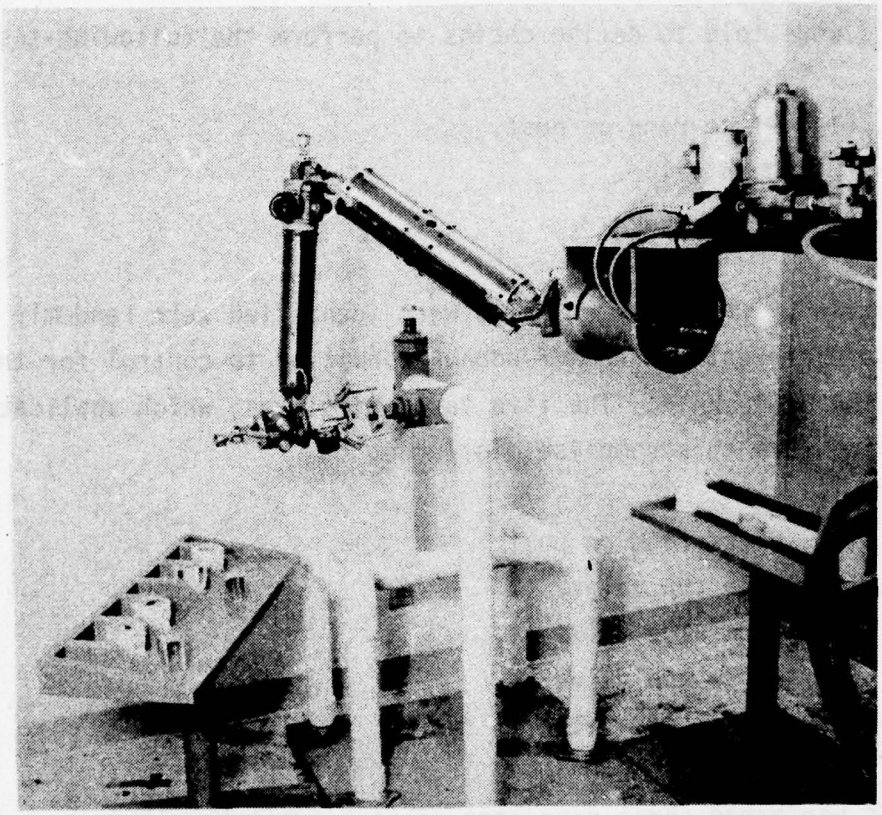


FIGURE 3-6.
VALVE TURNING TASK

"minimal feedback" on Session 2, and vice versa, with equal number of subjects receiving each feedback condition first.)

Subjects were told to define chains to perform the following tasks:

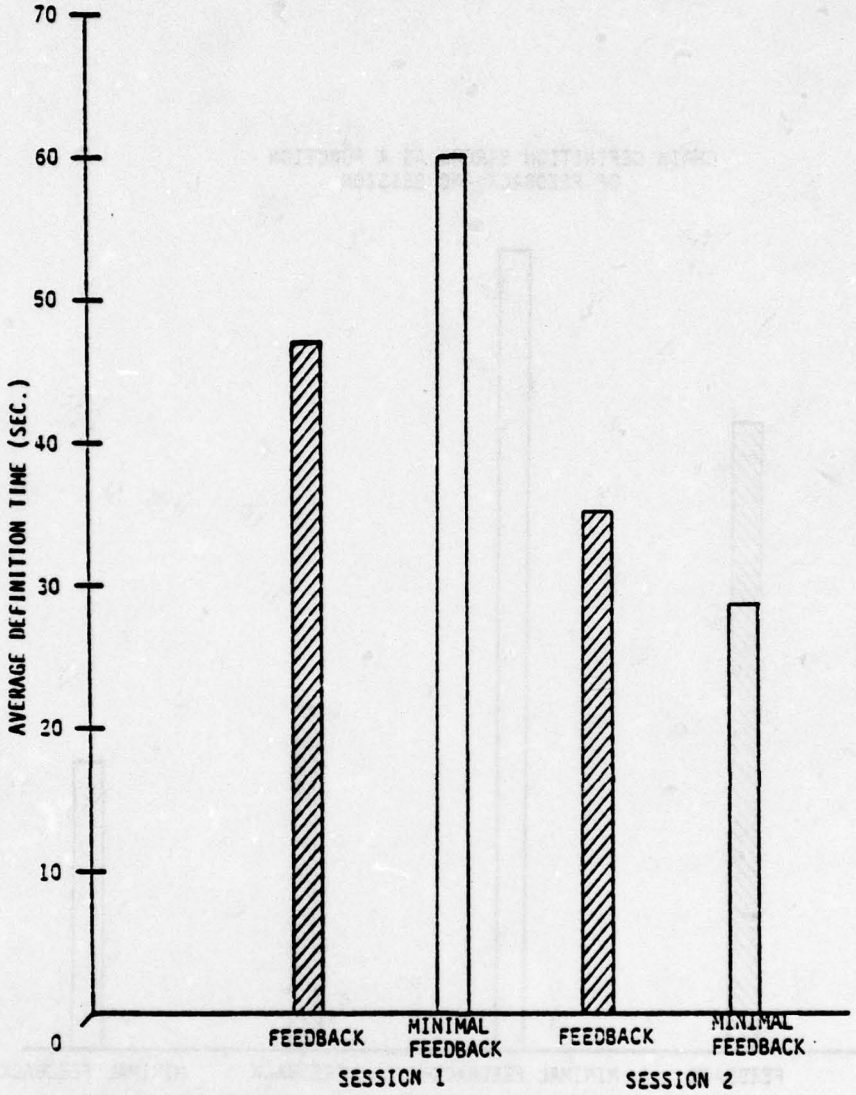
(1) Place ring on post.
(2) Open valve.
(3) Close valve.

The numbers by which these chains were identified were randomly assigned and were different for each feedback condition to control for the perseveration of learning. The five task objectives, which application of the chains were to accomplish, included:

(1) Place ring on post.
(2) Open top valve.
(3) Open side valve.
(4) Close top valve.
(5) Close side valve.

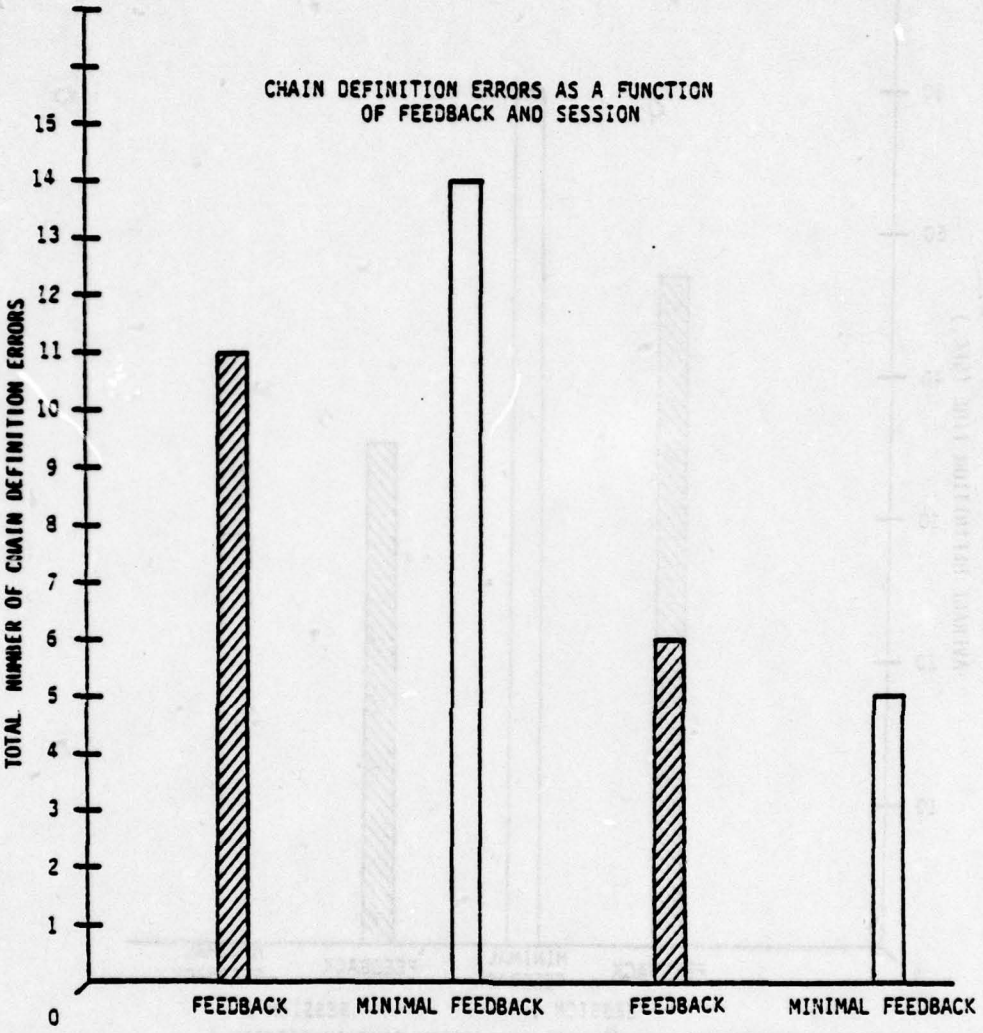
Subjects performed three sequences of five subtasks within each feedback condition. The order in which subtasks were to be performed was randomized, and the subtask to be performed at a given point in the sequence was identified by the experimenter after the completion of the preceding subtask. The time consumed and the number and type of errors committed in the performance of each subtask were recorded.

3.4.4 Results. The experimental results in terms of chain definition times and chain definition errors are shown in Figures 3-7 and 3-8. The results show that state feedback reduced both chain definition time and chain definition errors by more than 20% in the first session. However,



NOTE: WHILE FEEDBACK REDUCED DEFINITION TIME IN SESSION 1, THE EFFECT WAS NOT STATISTICALLY SIGNIFICANT.

FIGURE 3-7.
CHAIN DEFINITION TIME ON COMBINED RING PLACEMENT AND VALVE TURNING TASK.



NOTE: WHILE FEEDBACK REDUCED DEFINITION ERRORS IN SESSION 1, THE EFFECT WAS NOT STATISTICALLY SIGNIFICANT.

FIGURE 3-8.
TOTAL NUMBER OF CHAIN DEFINITION ERRORS AS A FUNCTION OF FEEDBACK CONDITION AND SESSION.

a large learning effect was observed between first and second session and feedback did not provide a similar advantage in the second session. A direct interpretation is that feedback aids only the construction of new chain by relieving the operator of any requirement to memorize the sequence of chain elements. The statistical tests indicated no significant effects of the feedback for overall chain definition times and errors.

The results of the average chain execution times and execution errors under two feedback conditions and three consecutive trials are shown in Figures 3-9 and 3-10. The analysis of variance for chain execution time, summarized in Table 3-2, shows that the effects of session ($P < 0.01$) were significant. The execution time and execution error differences between feedback conditions were not significant, although performance improvement in later sessions seems to be greater in full feedback conditions than in minimal feedback conditions. This confirmed the subjects' self-observation of infrequent attendance to the screen when the visual display of previously-defined chain components was presented in the "full feedback" conditon during chain execution period.

There were three major reasons which apparently contributed to the lack of significant differences between feedback conditions. First, it was expected that due to the visual richness of the two-view, non-degraded TV feedback, the effect of machine feedback observed would be secondary, if at all discernible. Second, the task used in the experiment was relatively simple and familiar to the subjects with previous training experience. It is the high variability components rather than the high repetition components of the task that are expected to continuously demand cognitive monitoring of the operator in a supervisory situation. Third, the actual difference between the two feedback conditions was not as distinct as their names

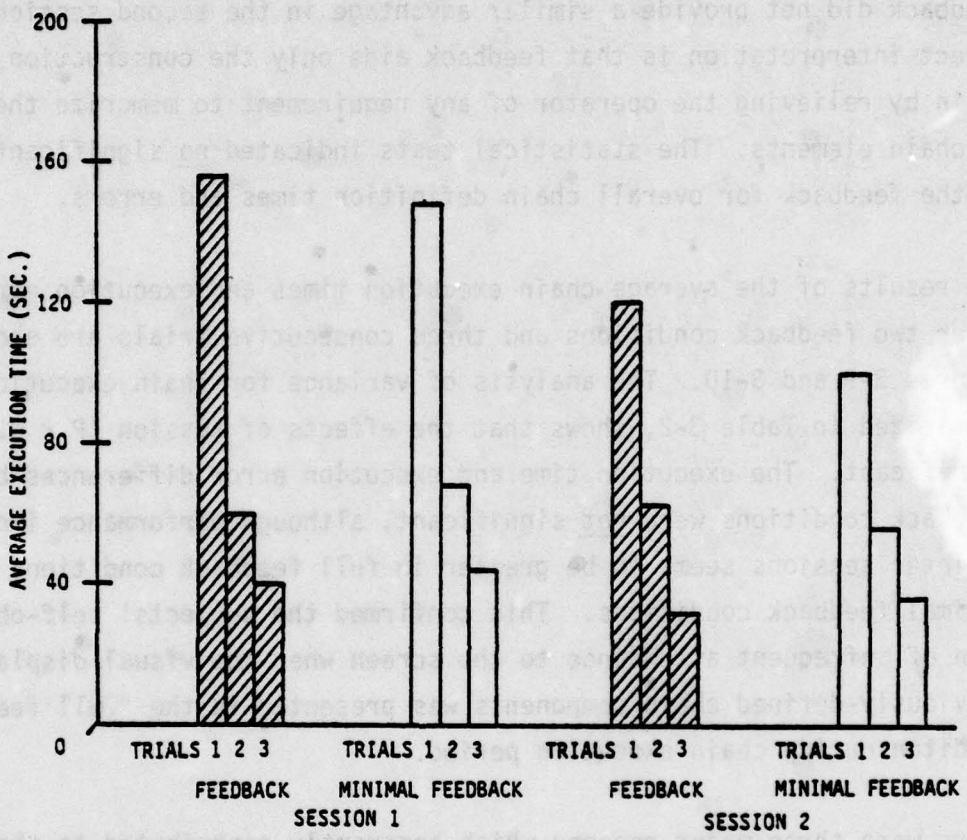


FIGURE 3-9.
CHAIN EXECUTION TIME ON COMBINED RING PLACEMENT AND VALVE TURNING TASK.

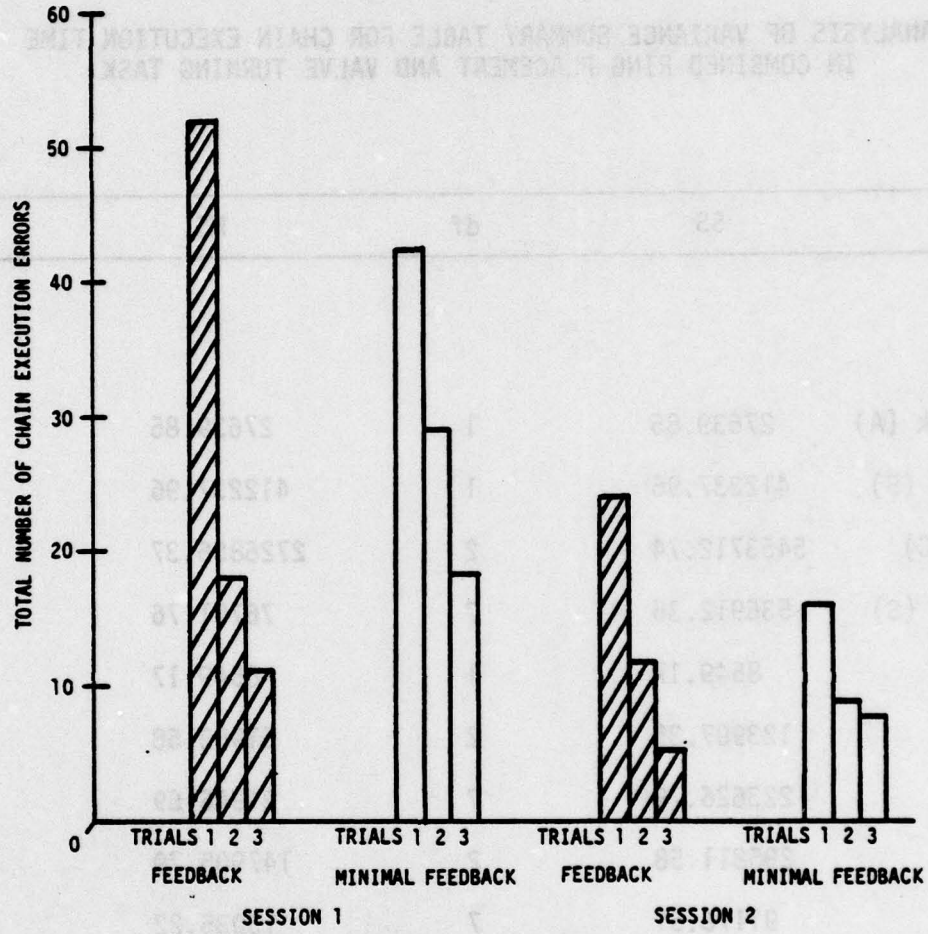


FIGURE 3-10.
TOTAL NUMBER OF CHAIN EXECUTION ERRORS

TABLE 3-2

ANALYSIS OF VARIANCE SUMMARY TABLE FOR CHAIN EXECUTION TIME IN COMBINED RING PLACEMENT AND VALVE TURNING TASK

Source	SS	df	MS	F
State Feedback (A)	27639.85	1	27639.85	< 1
Session (B)	412237.96	1	412237.96	31.65*
Trial (C)	5453712.74	2	2726856.37	46.68*
Subject (S)	536912.35	7	76701.76	
A×B	8549.17	1	8549.17	< 1
A×C	123907.35	2	61953.68	1.32
A×S	223626.79	7	31946.69	
B×C	295811.58	2	147905.79	3.74**
B×S	91176.51	7	13035.22	
C×S	817850.00	14	58417.89	
A×B×C	75311.06	2	37655.53	1.11
A×B×S	317573.63	7	45367.66	
B×C×S	553477.55	14	39534.11	
A×C×S	657384.93	14	469556.06	
A×B×C×S	475817.43	14	33986.96	

* $P<0.01$

** $P<0.05$

imply. On one hand, the "minimal feedback" condition omitted visual display of all previously defined chain components but did present a display of the currently operative command, including both visual and auditory signal of the "manual" function, so that the operator was never unaware of the requirement to perform a manual response. On the other hand, the feedback format and structure were system-oriented (a queue listing of subtasks), rather than user-oriented (hierarchically structured schematic diagram form, for example). Further research might devote itself to a refinement of the machine state feedback conditions to permit analysis of relationship between feedback specificity and operator performance.

3.5 Experiment 3: Automatic Command

The purpose of this third experiment was to evaluate the SMC^3 language features in different task environments. Within the current test environment, a six-degree-of-freedom, wrist force sensor was inserted between the manipulator arm and the gripper device (Figure 3-11). Due to the sensor hardware deficiency, the implementation of control software for interactive computer control based on the processed sensor information included only:

(1) Automatic monitoring and reporting (in the presence of detected force/torque on end effector).

(2) Initiation of the "stop and hold" routine, along with a warning tone at the operator station in the event of force/torque exceeding designated threshold. In automatic command mode, chain under execution is aborted and the control switches to manual mode.

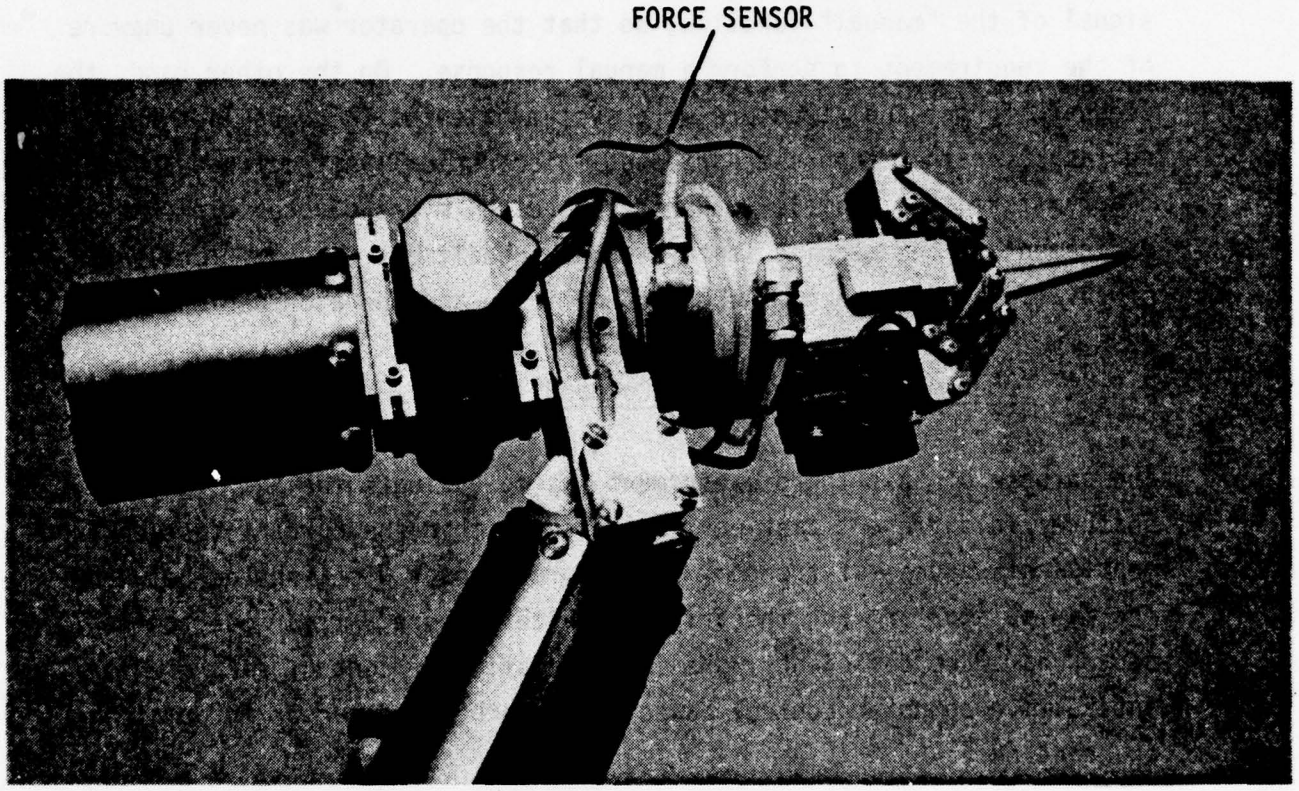


FIGURE 3-11.
WRIST FORCE SENSOR UNIT

Another dimension of this experiment was visibility simulation which represented combined attributes of TV monitors, camera, lighting and water properties. The net result was the quality of TV image presented to the operator. A simplified, two-visibility level was thus established through the contrast and brightness controls of the monitors. Under normal viewing conditions, cameras and monitors were adjusted for the best presentation of the manipulation and work space. The luminance levels of white and black calibration squares were held at approximately 35 and 9 candelas/m^2, respectively, measured with a Pentax 1° angle spot-meter. This corresponded to a 59% contrast ratio. Under degraded viewing conditions, the luminance levels were held at approximately 4.5 and 3.2 candelas/m^2 with a moderate 17% contrast ratio. This procedure enabled a gross representation of the visibility factor in our study of man-computer communication in controlled manipulation.

A modified machine feedback display was used in this experiment. As shown in Figure 3-12, the list of top-level commands is statically displayed in the left portion of the screen. A pointer moves down the list as execution of these commands progresses. When execution of last command is completed, the whole list of commands is then erased and replaced by another list. This new form of display seems to provide easier perception of machine state during execution than the previous design, based on the subject's opinion.

3.5.1 Practice. Prior to the experimental sessions, six participants underwent several hours of orientation and practice. This unstructured practice included:

(1) Basic manipulation.
(2) Task breakdown and language commands.
(3) Two-view TV viewing under both normal and degraded conditions.

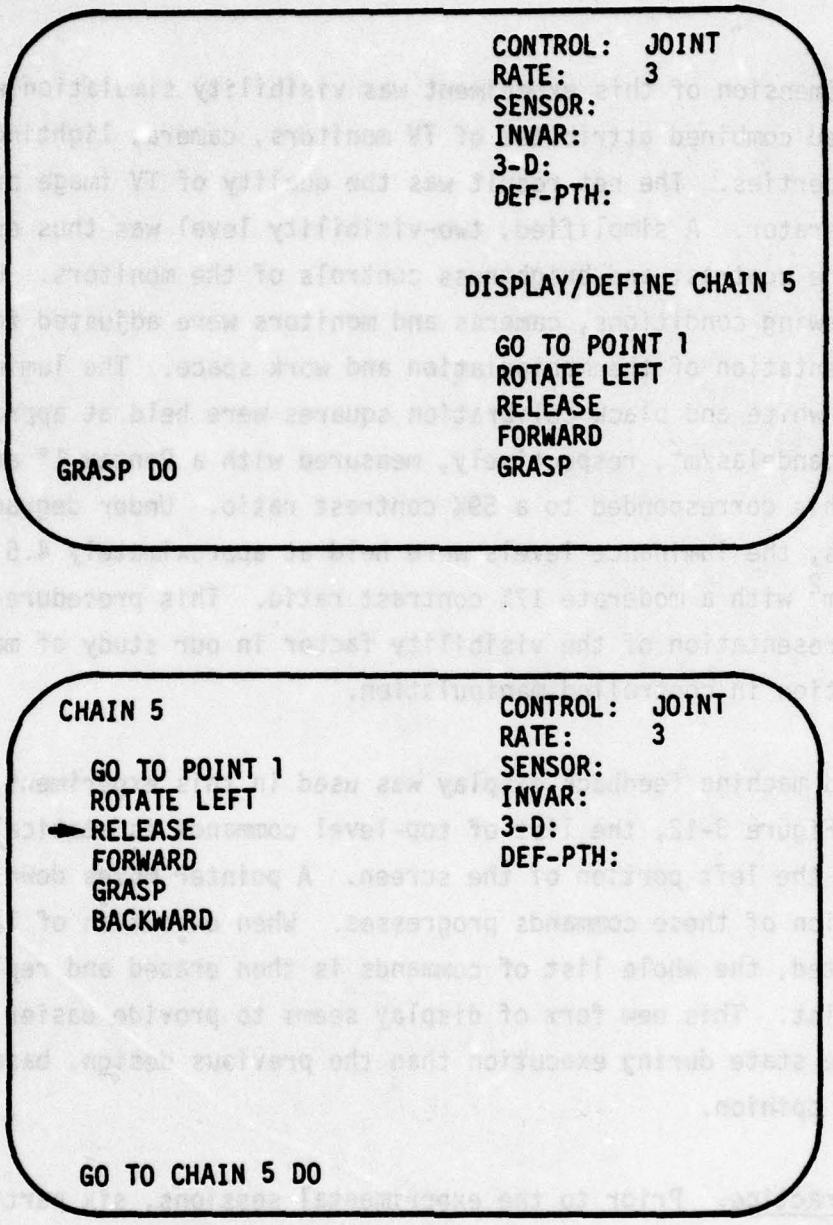


FIGURE 3-12.
IMPROVED MACHINE STATE FEEDBACK DISPLAY DURING CHAIN DEFINITION (TOP) AND DURING CHAIN EXECUTION (BOTTOM)

3.5.2 Task. In the experiment, the participants were instructed to use variable and automatic commands to perform a simulated maintenance task under both normal and degraded viewing conditions. A 2 x 2 repeated measures design was used to exmaine the effects of command modes and visibility.

The simulated maintenance task was based on a pipe structure adopted from last year's design. It consisted of three main subtasks: (1) valve closing, (2) cap replacements, and (3) valve opening. The task setup is shown in Figure 3-13. It included two gate "valves" with square handles, two "caps," a pipe sections, and a "table."

To complete the maintenance task, the participants were required to perform the following steps:

(1) Go to top valve.
(2) Turn the valve off (rotate it five times clockwise).
(3) Go over the cap in crossbar.
(4) Grasp the cap and remove it from crossbar.
(5) Place cap on the table.
(6) Pick up new cap on the table.
(7) Transport over crossbar.
(8) Place cap in crossbar.
(9) Go over top valve.
(10) Turn the valve on (rotate it five times counterclockwise).
(11) Return to stow position.

3.5.3 Procedure. Prior to the experimental sessions, the participants read the instructions. A task list, which listed all of the required task steps in the sequence in which they were to be performed, was taped to the work station. Before each session, the participants were given

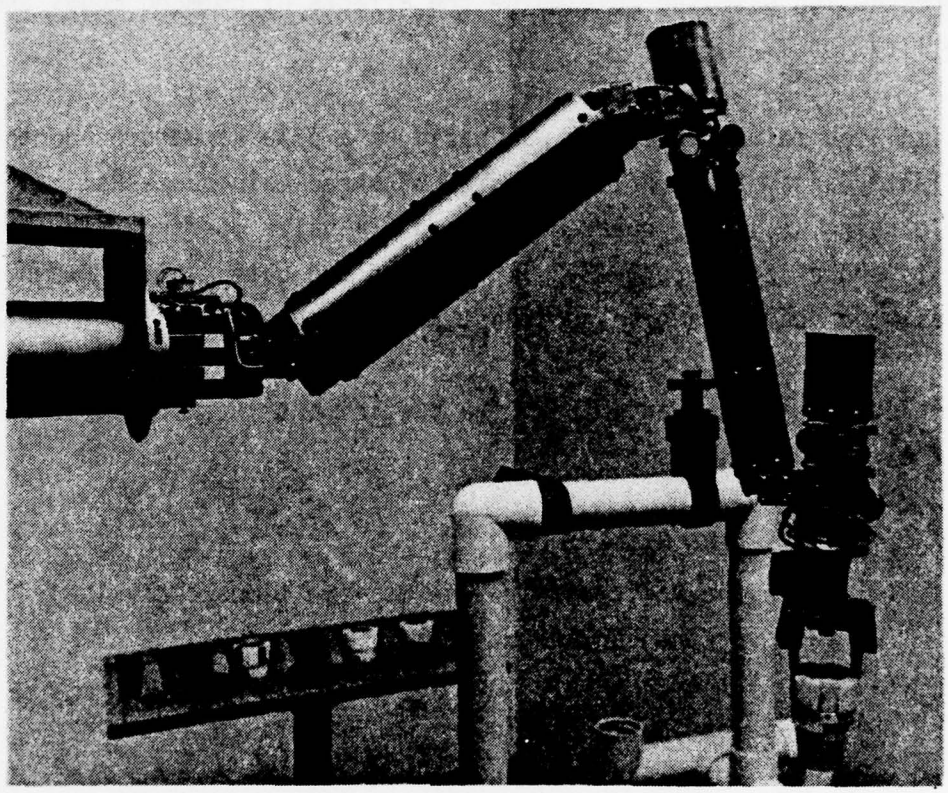


FIGURE 3-13.
SIMULATED MAINTENANCE TASK SETUP

several minutes to plan their strategy for accomplishing the overall task and to consult with the experimenter for the feasibility of the planned approach. The point/path assignment period then followed, in which the participants were allowed to assign up to ten spatial points and two paths for later usage in any manner they desired. If the chained command mode followed, the participants started with the chain definition period. They were allowed to assign up to five chains. The normal pattern was to assign one chain for each subtask and to summarize the main task into another chain which included:

(1) Go to position over top valve.
(2) Go to chain 3.
(3) Go from position by side valve to stow.

Where chain 3 consists of chain 0, 1, and 2, which performed valve closing, valve opening, and cap replacement subtasks, respectively.

If the fixed command mode followed, the participant started to perform the tasks, using all the basic commands available except the chaining capability. The tasks were performed in a continuous manner, with the experimenter recording various task errors on the experimental data sheet. The time history as well as command/control errors were then recorded by the automatic data recording routine.

3.5.4 Results. This section discusses the experimental results based on the data analysis of task times, task errors, and control strategies employed. The analysis is presented both for overall task and subtask elements. The implication and extension of these results are discussed in the next section.

Overall Task Time. Table 3-3 presents the time required for each participant to command and execute the entire task under each of the four test situations. The results of a within-subject analysis of variance on task execution time (Table 3-4) indicated that participants performed significantly better (faster and more consistently) using Chained commands than they did using Fixed commands ($F = 52.66$, $P < 0.001$). This represents a 56% reduction in execution time with the normal visibility and a 60% reduction with the degraded visibility than with the normal visibility ($t = 2.97$, $P < 0.05$). No significant main visibility effect was obtained and the two-way interaction (command x visibility) was not statistically significant ($P > 0.05$).

The average task completion time data are graphically illustrated in Figure 3-14. As shown in fixed mode performance, it took an average of 17% longer for the participants to complete the task with the degraded TV than with the clear TV ($t = 2.61$, $P < 0.025$). However, this degradation had more than been compensated by the use of Chained commands with which the participants were able to reduce task completion time by 18% ($t = 2.76$, $P < 0.025$). An important conclusion drawn from the preceding was that the Chained commands had effectively reduced the detrimental effects of vision conditions based on overall task completion time.

Separate task performance analyses were conducted for two major subtasks to provide a comprehensive description of the contribution of each subtask to total task performance. These analyses were based on the assumption that the performance measures in individual subtask command and and execution (e.g., completion times and errors) are additive and the interactions among them are negligable.

Valve Turning. The participant performance time for valve turning task (Step 2 and 10) are calculated and summarized in Table 3-5. An analysis

TABLE 3-3

PARTICIPANT OVERALL TASK PERFORMANCE (SEC)
AS A FUNCTION OF VISIBILITY AND COMMAND MODE

	CONDITION								
	CLEAR TV				DEGRADED TV				
	(A)	(B)			(C)	(D)			
SUBJECT	FIXED	CHAINED			FIXED	CHAINED			
		DEFINITION	EXECUTION	TOTAL		DEFINITION	EXECUTION	TOTAL	
1	574	329+	150‡ (309)	788	801	228+	138‡ (254)		624
2	781	371	124 (422)	917	723	304	82 (328)		714
3	1123	442	109 (348)	899	1238	351	156 (535)		1042
4	761	278	149 (427)	854	1044	238	116 (355)		709
5	707	156	89 (295)	540	863	295	80 (415)		790
6	575	258	138 (197)	593	632	250	123 (222)		595
MEAN	753.50		(333.00)	765.17	883.50		(351.50)		745.67
S.D.	201.79		(86.60)	161.07	222.56		(113.63)		161.02

+ Point/path definition
‡ Chain definition

TABLE 3-4

ANALYSIS OF VARIANCE SUMMARY TABLE FOR
OVERALL TASK EXECUTION TIME

SOURCE	df	MS	F
Command Mode (C)	1	1208708.16	52.66**
Visibility (V)	1	10416.66	1.4
Subject (S)	5	66964.34	
C x V	1	8437.51	3.42
C x S	5	22952.07	
V x S	5	7451.30	
C x V x S	5	2467.60	

** $p < 0.001$

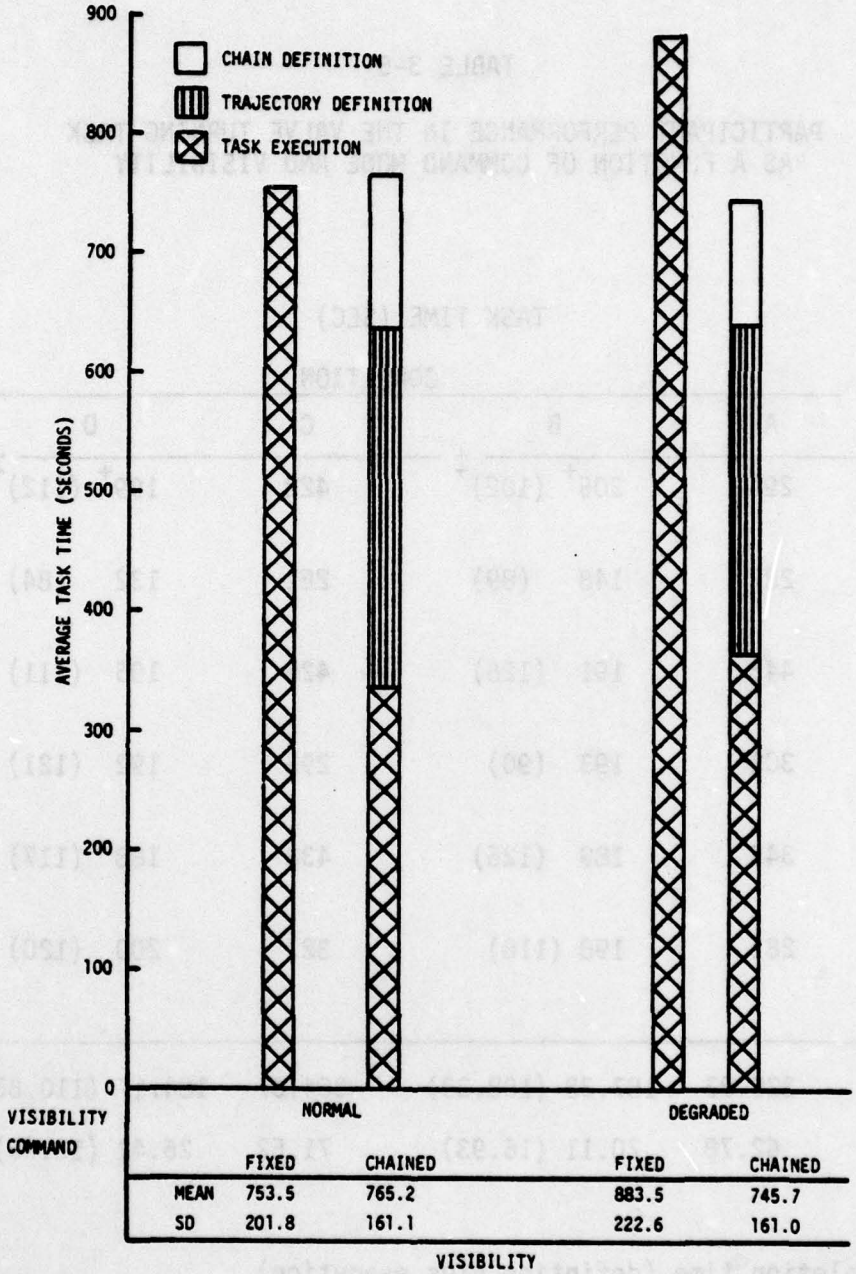


FIGURE 3-14.
AVERAGE TASK COMPLETION TIME AS A FUNCTION OF COMMAND MODE AND VISIBILITY

TABLE 3-5

PARTICIPANT PERFORMANCE IN THE VALVE TURNING TASK
AS A FUNCTION OF COMMAND MODE AND VISIBILITY

TASK TIME (SEC)

	CONDITION			
SUBJECT	A	B	C	D
1	294	205+ (102)‡	422	199+ (112)‡
2	287	148 (89)	281	132 (84)
3	445	191 (126)	425	195 (111)
4	302	193 (90)	295	192 (121)
5	346	189 (125)	438	188 (117)
6	281	198 (118)	327	200 (120)
MEAN	325.83	187.33 (108.33)	364.67	184.17 (110.83)
S.D.	62.78	20.11 (16.93)	71.52	26.42 (13.76)

+ Task completion time (defintion plus execution)

‡ Task execution time

of variance was performed to confirm the significant improving effects of the command mode ($F = 10.93$, $P < 0.025$). The effect of visibility was found to be minimal. We believe this is due to two major factors related to task characteristics. First, the level of visual activity and hand-eye coordination required to perform the valve turning task was low. Second, this task consisted of a structured group of primitive elements with several fail-safe repetitions. As an indicator for the advantage of Chain command, the savings in execution time and the overhead in pre-program (definition) time were calculated. The ratios of benefit over cost in valve turning task were 2.75 in normal TV and 3.45 in degraded TV.

Cap Replacement. The participant performance times for cap replacement (task Step 3 through Step 8) are calculated and summarized in Table 3-6. Neither of the main effects (command or visibility) reached a generally acceptable level of significance ($P > 0.05$) based on an analysis of variance of task execution time. However, in Fixed command mode, in which manual joystick control is the primary mode of control employed by the participants, a significant 43% degradation in performance time was found in degraded TV compared to normal TV ($t = 3.04$, $P < 0.025$). The benefit/cost ratios for Chained commands were calculated to be .75 (normal) and 1.26 (degraded) for cap replacement task.

Task Errors. Figure 3-15 summarizes the errors committed in the execution of the integrated maintenance task. An analysis of execution errors, shown in Table 3-7, indicated that participants made significantly fewer errors with Chained commands than with Fixed commands ($F = 7.12$, $P < 0.05$). While normal TV results in less errors, the difference is not statistically significant. Approximately two-thirds of task errors were committed in the cap replacement task which called for precise position control and multi-degree-of-freedom motion. Most of the errors committed in valve turning were eliminated with the use of Chained commands, while those committed in cap replacement also were reduced significantly (see Table 3-8).

TABLE 3-6

PARTICIPANT PERFORMANCE ON THE CAP REPLACEMENT TASK
AS A FUNCTION OF COMMAND MODE AND VISIBILITY

TASK TIME (SEC)

	CONDITION			
SUBJECT	A	B	C	D
1	176	250+ (142)‡	251	190+ (39)‡
2	296	428 (188)	321	302 (131)
3	351	280 (128)	528	527 (341)
4	362	388 (231)	626	365 (175)
5	215	210 (100)	279	390 (214)
6	143	228 (33)	209	217 (63)
MEAN	257.16	297.33 (137.00)	369.00	331.83 (160.50)
S.D.	92.42	89.74 (68.89)	168.08	123.91 (110.26)

+ Task completion time
‡ Task execution time

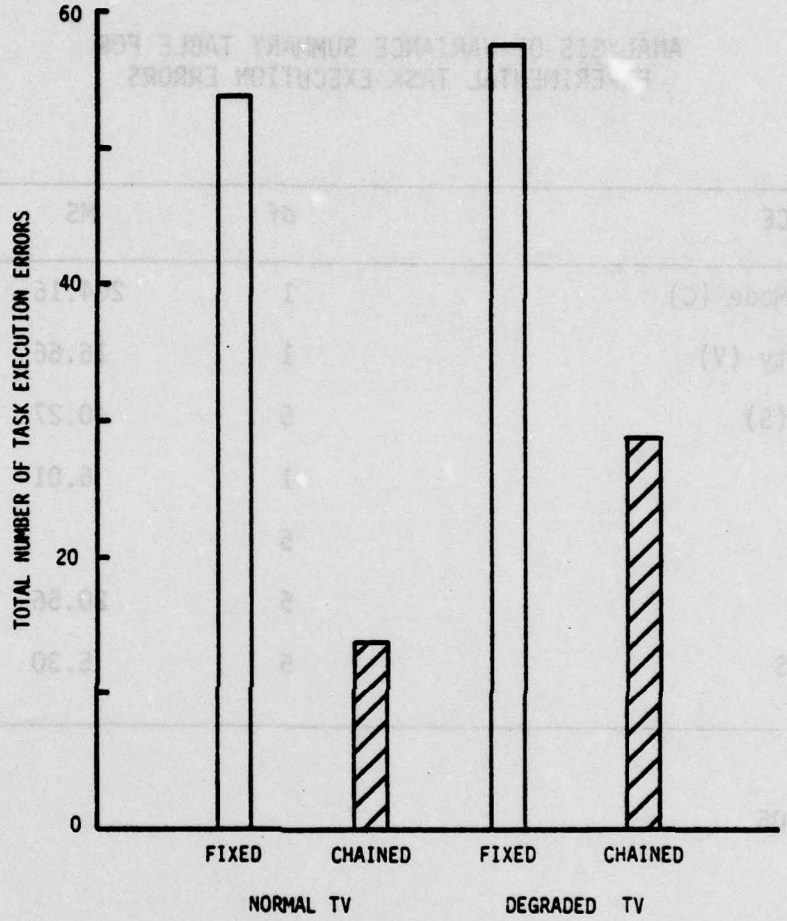


FIGURE 3-15.
TOTAL NUMBER OF MAINTENANCE TASK EXECUTION ERRORS AS A FUNCTION OF COMMAND AND VISIBILITY. NOTE THAT WHILE NORMAL VIEWING RESULTS IN BETTER PERFORMANCE, THESE DIFFERENCES ARE NOT STATISTICALLY SIGNIFICANT

TABLE 3-7

ANALYSIS OF VARIANCE SUMMARY TABLE FOR
EXPERIMENTAL TASK EXECUTION ERRORS

SOURCE	df	MS	F
Command Mode (C)	1	204.16	7.12*
Visibility (V)	1	16.66	<1
Subject (S)	5	40.27	
C x V	1	6.01	1.13
C x S	5		
V x S	5	20.56	
C x V x S	5	5.30	

* $p < 0.05$

TABLE 3-8

SUBTASK COMPARISON SUMMARY TABLE

Task Characteristics	Valve Turning	Cap Replacement
. Repetition	4	½ (repeated in reverse order)
. Trajectory Complexity	2-degree-of-freedom	mixed multi-degree-of-freedom
. Motion Requirements	Short distance unit motion	medium distance transport plus short distance positioning
. Accuracy Requirements	Gross (1" allowance)	Fine (¼" allowance)
. Environmental Interaction	Low contact (visual & force)	High contact
Command Features		
. Command Automation	Symbolic-oriented	Analogic-oriented
. Command Uniformity	High (fixed)	Low (Variable)
. Control Transfer	0	2 to 3
. Chaining Level	1	1 to 2
. Feedback Update	Fast ($\sim 1^{sec}$/entry)	slow ($\sim 4^{sec}$/entry)
. Feedback Usage (Mode selection, force message)	Very Few	Some
Performance Measures		
. Visibility Degradation (% time)	12%+	43%*
. Command Improvement (% time)	49%*	12%+
. Command Benefit/ Definition cost (clear) (degraded)	2.75 3.45	.75 1.26
. Task Execution Error (Manual) (Automatic)	 37 2	 52 21
. Manual Control Allocation	insignificant	76% time

* statistically significant at 0.05 level

\+ did not reach acceptable level of significance

Although there were a few errors committed in definition periods, the overall trends were not affected.

Table 3-8 compares subtask characteristics, command features used, and resulting performance for each type of subtask. As shown, the valve turning subtask has a potentially high payoff value for the application of Chained commands, evidenced by its high task time reduction (49%), high benefit/overhead ratio (between 3.75 and 3.45 for the cases tested), and significant error reduction. This we believe is due to appropriate command usage in a task of a repetitive and structure nature, which is further characterized by trajectory complexity, accuracy and environment requirements.

Control Allocation. To illustrate typical command and control input samples, Figure 3-16 plots the computer-sampled joystick and keyboard command input versus time for two experimental runs (one with the Fixed command and one with the Chained command). It can be seen that discrete control actions were adopted by the participant and the intensity of control effort may be appropriately indicated by the frequency and duration of the control action. It appears that a skillful participant employs a "control-and-wait" strategy while he is in manual control. As an example, manual cap insertion activities of the participants can be summarized as a sequence of joystick control periods each averaged 4.6 seconds and preceded by a 0.6-second "waiting" time. No significant pattern shift was found in control transfer (minimal vs. automatic) and between viewing conditions; although participants tend to adopt longer "wait" time (1.6 seconds) in degraded TV viewing condition.

A summary of command type frequencies from recorded keyboard entries indicated that the Chained command had significantly reduced the use of analogic commands and control primitives including FORWARD, BACKWARD,

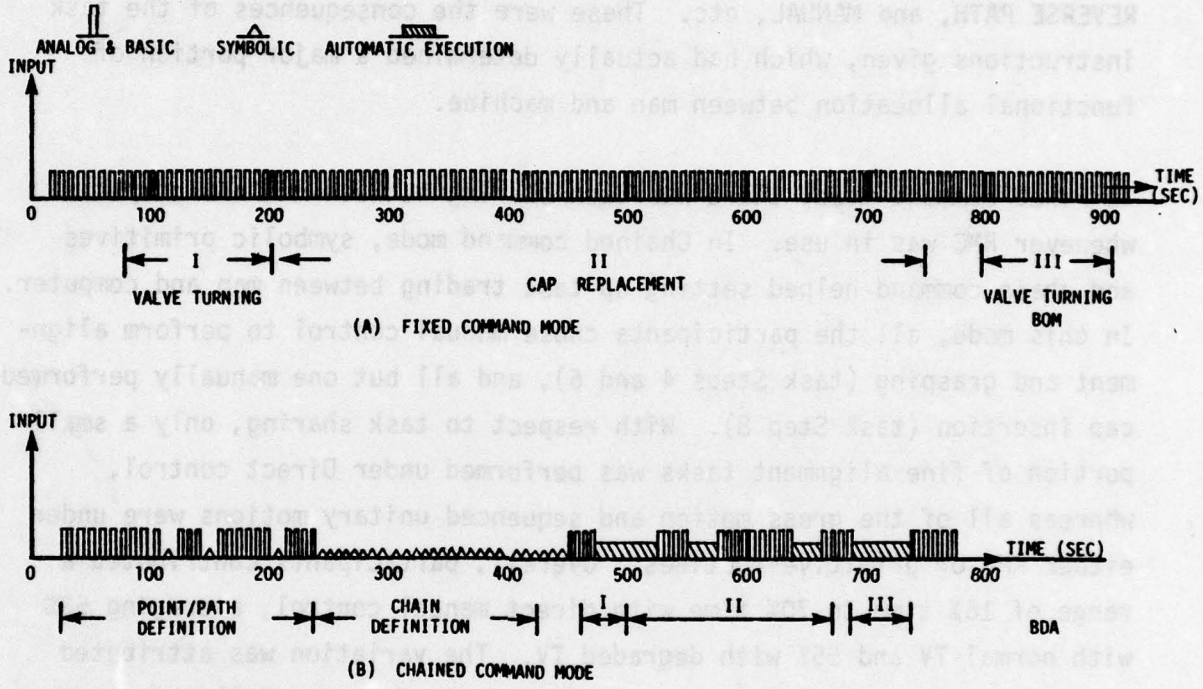


FIGURE 3-16.
COMMAND AND CONTROL INPUTS DURING FIXED COMMAND MODE (TOP) AND CHAINED COMMAND MODE (BOTTOM)

GRASP, RELEASE, ROTATE RIGHT, and ROTATE LEFT. It had, however, increased the uses of mode selection (typically, motion RATE, and RMC/Direct switches), and various variable primitives such as GO TO, DEFINE, CHAIN, POINT, PATH, REVERSE PATH, and MANUAL, etc. These were the consequences of the task instructions given, which had actually determined a major portion of functional allocation between man and machine.

In Fixed command mode, there was task sharing between man and computer whenever RMC was in use. In Chained command mode, symbolic primitives and chain command helped setting up task trading between man and computer. In this mode, all the participants chose manual control to perform alignment and grasping (task Steps 4 and 6), and all but one manually performed cap insertion (task Step 8). With respect to task sharing, only a small portion of fine alignment tasks was performed under Direct control, whereas all of the gross motion and sequenced unitary motions were under either RMC or primitive routines. Overall, participants contributed a range of 16% time to 70% time with direct manual control, averaging 53% with normal TV and 55% with degraded TV. The variation was attributed to individual participant's training and aptitude, especially his trust in the machine, and his capability to identify a "reach-align" type task breakdown in unit level.

The basic decision encountered by the operator in using chained commands had been: Should a specific subtask element be traded under computer control? Based on the operator interview, several factors were considered in the tradoffs involving time and efforts spent in definition, execution, and control transfer. These included position and force accuracies and speed criticality perceived by the operator in his task-tool match. Hence, operator confidence in traded control with SMC^3L is a function of task-tool match, capability in task breakdown, state feedback adequacy, and training.

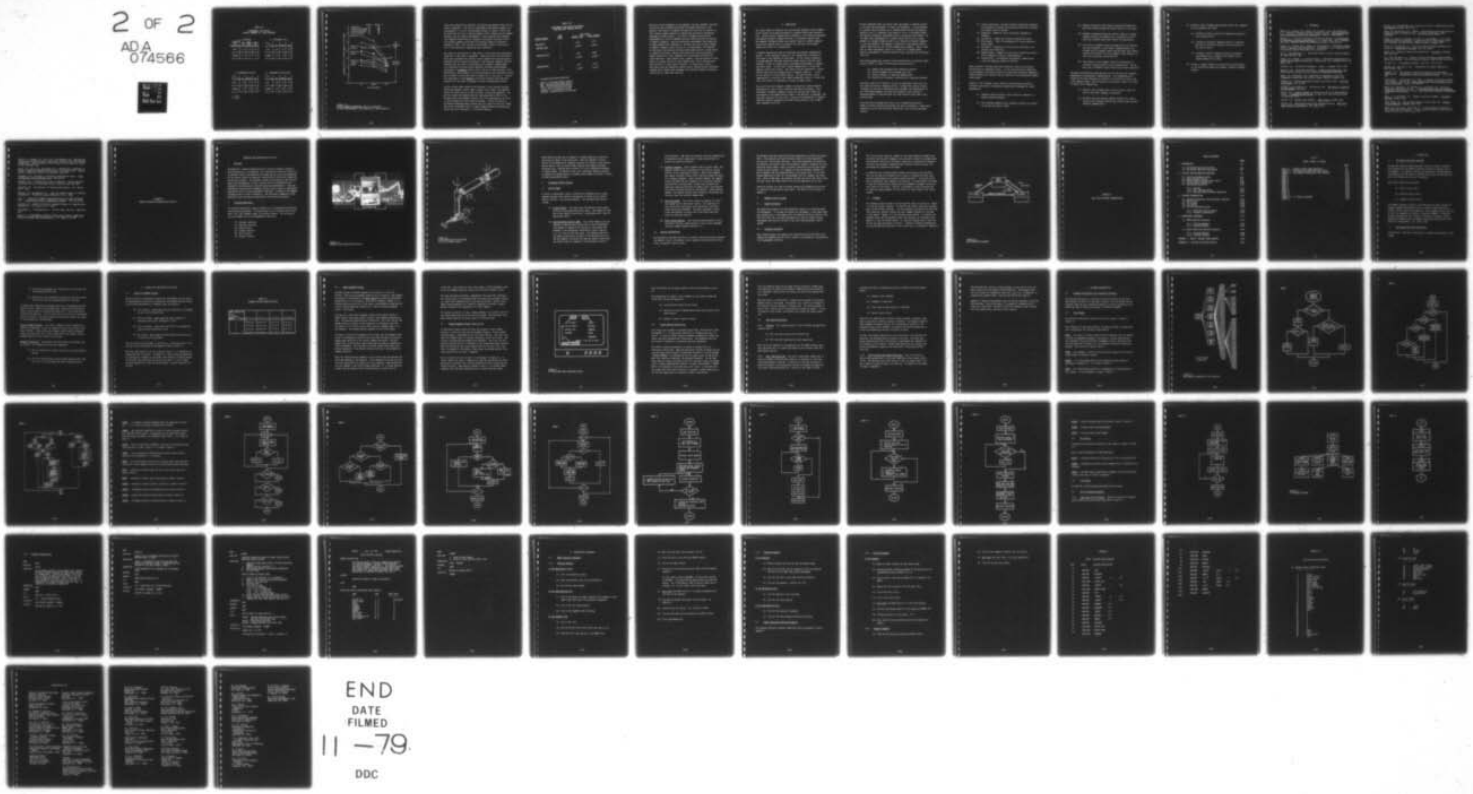
To summarize the results of performance time evaluation, the Chained/Fixed command performance time ratios were calculated and compared for cap replacement, overall task, and valve turning task. As shown in Table 3-9A, performance time ratios vary with environmental (visibility) factor and intrinsic task nature. The later factor is largely attributable to the discrete and repetitive nature of the task; and the tasks listed above form a natural rank order of task structuredness. We found that further gains in speed were significant when some levels of preprogramming were allowed. This is another advantage of traded man-computer control in remote manipulation, especially when environmental uncertainties can be reduced and tasks are well specified beforehand. The performance ratios for three levels of possible prespecification, i.e., pre-defined chains, pre-defined trajectory, and pre-defined chains/trajectory, were calculated and shown in Tables 3-9B, C, and D. The biggest gains are obtained when both task procedure and task geometry are preprogrammed as expected, followed with preprogrammed point/path. The data presented in this table are a start toward the relative prediction of performance level usable in the selection of the best control mode (Chained vs. Fixed, or Traded vs. Shared) from a time standpoint. The performance ratios are further plotted as a function of task structuredness, task specificity and visibility in Figure 3-17. The high-payoff application of the SMC^3L-type traded control would be at the lower right area where tasks are highly structured, clearly specified for preprogramming, and of low visibility.

3.6 Discussions

Two comparisons of the present data with those collected in the previous years are of interest. First, the time performance of various command/control modes were compared. Since the command language implementation and evaluation were conducted in a stepwise approach and the experimental

AD-A074 566 PERCEPTRONICS INC WOODLAND HILLS CALIF F/G 5/8
MAN-MACHINE COMMUNICATION IN COMPUTER-AIDED REMOTE MANIPULATION--ETC(U)
MAR 79 W H CROOKS, E SHAKET, Y CHU N00014-76-C-0603
UNCLASSIFIED PATR-1034-79-3 NL

2 OF 2
AD A 074566



END
DATE FILMED
11 -79
DDC

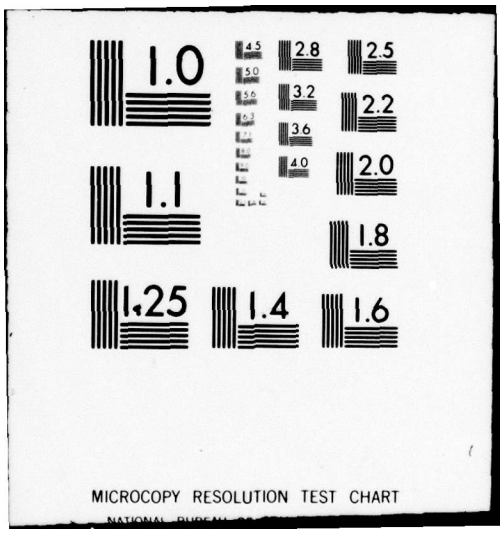

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART

TABLE 3-9

PERFORMANCE TIME RATIOS: CHAINED COMMANDS VS. FIXED COMMANDS

(A) NO PREPROGRAM

STRUCTURED-NESS / VISIBILITY	CAP TASK LOW	OVERALL TASK MEDIUM	VALVE TASK HIGH
CLEAR	1.16	1.02	.57**
DEGRADED	.90	.84*	.51**

(B) PREPROGRAMMED CHAINS

S / V	LOW	MEDIUM	HIGH
CLEAR	.98	.85	.46**
DEGRADED	.76*	.71**	.41**

(C) PREPROGRAMMED POINT/PATHS

S / V	LOW	MEDIUM	HIGH
CLEAR	.71*	.61*	.45**
DEGRADED	.57*	.53**	.40**

(D) PREPROGRAMMED POINT/PATH/CHAINS

S / V	LOW	MEDIUM	HIGH
CLEAR	.53**	.44**	.33**
DEGRADED	.43**	.40**	.30**

* $P<0.05$

** $P<0.005$

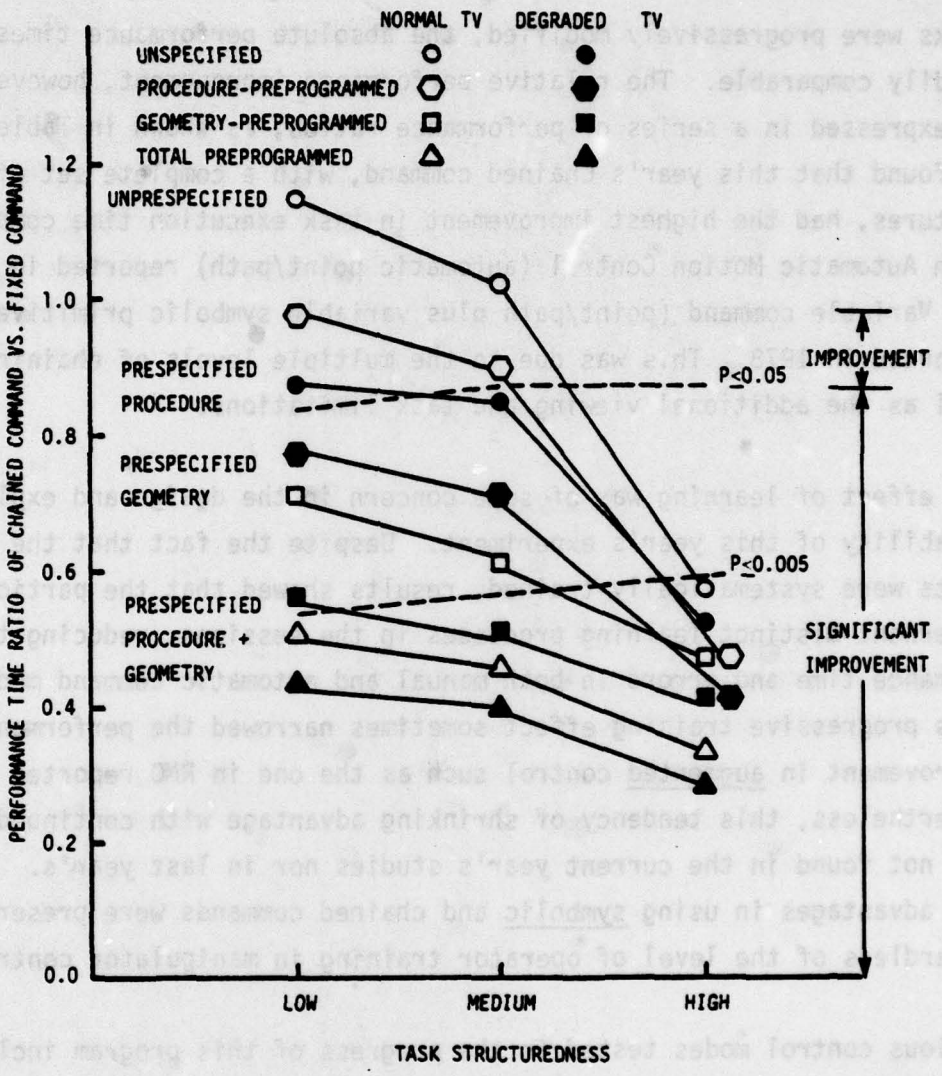


FIGURE 3-17.
CHAINED-COMMAND PERFORMANCE RATIO AS A FUNCTION OF TASK STRUCTUREDNESS, TASK SPECIFICITY, AND VISIBILITY

tasks were progressively modified, the absolute performance times are not readily comparable. The relative performance improvement, however, can be expressed in a series of performance ratios, as shown in Table 3-10. We found that this year's Chained command, with a complete set of command features, had the highest improvement in task execution time compared with Automatic Motion Control (automatic point/path) reported in 1977, and Variable command (point/path plus variable symbolic primitives) reported in 1978. This was due to the multiple levels of chaining as well as the additional viewing and task limitations.

The effect of learning was of some concern in the design and exploratory capability of this year's experiment. Despite the fact that the participants were systematically trained, results showed that the participants underwent distinct learning processes in the sessions, reducing task performance time and errors in both manual and automatic command modes. This progressive training effect sometimes narrowed the performance improvement in augmented control such as the one in RMC reported in 1977. Nevertheless, this tendency of shrinking advantage with continued training was not found in the current year's studies nor in last year's. Instead, the advantages in using symbolic and chained commands were preserved, regardless of the level of operator training in manipulator control.

Various control modes tested in the progress of this program included shared man-computer control (RMC), traded control (AMC), and combined shared and traded control (Fixed, Variable, and Chained commands). Our earlier studies had demonstrated the speed advantage of RMC in line following and ring placement subtasks, but this advantage was not found in fine alignment and valve turning subtasks. Similar variation in performance advantage among subtasks was also found with the use of AMC. The overall performance gains in these computer-control functions were simply the sum of the gains in individual subtasks.

TABLE 3-10

PERFORMANCE TIME RATIOS FOR OVERALL TASK AND VALVE TURNING SUBTASK

CONTROL MODES*	TEST PHASE	TIME RATIOS OVERALL TASK	VALVE TURNING
RMC/DIRECT	1	0.91	0.99
(AMC+RMC)/RMC	1	(0.98)+	(0.93)+
VARIABLE/FIXED	2	0.95	0.88
	2	(1.08)+	(0.95)+
CHAINED/FIXED	3	0.61	0.45
	3	(1.02)+	(0.57)+

+ Inlcuding Point/Path Definition

* Direct - Fixed Rate Manual Control
RMC - Resolved Motion Rate Control
AMC - Recorded Point/Path Motions
Fixed - Basic Primitives Plus RMC
Variable - Point/Path Plus Fixed
Chained - Chain-In-Chain Plus Variable

With more control delegated to the computer, and more automatic functions (primitives and variables) and control selections available to the operator, the effects of command structure and provision for smooth control transfer become important in determining the overall performance advantage of traded control. The effect of command structure was represented by a level of command construct from primitives and the use of symbolic commands. Relative performance improvements were found with a higher command level (Chained vs. Variable) and with a proper linkage of symbolic commands (valve turning vs. cap replacement, for example). In both cases, the grouping of established traded controls, either chains or symbolic primitives, required less definition time and resulted in higher benefit/cost ratio, compared with the grouping of manual and shared functions. Many complex tasks, however, required grouping of both manual and traded controls at some command level, which incurred extra time and effort in each control transfer. In these cases, control efficiency could be improved by smooth control transfer and by structured commands which promote task breakdown, well-defined automatic functions, and the use of high-level chaining.

4. CONCLUSIONS

This study sought to determine the relationships between critical communication factors and system performance in computer-aided remote manipulation. Current results demonstrate the benefits in utilizing well-designed, user-defined automatic command functions in performing complex tasks. The feedback information about the internal state of a complex computer control with complex tasks is essential for efficient and confident supervisory control of a manipuliator.

A stepwise approach was used to investigate man-computer communication factors. A few elementary aspects of the language were introduced at each stage in the sequence of experiments. As the studies progressed, it became apparent that both shared and traded control functions were required in order to achieve overall performance improvement in complex, real-world tasks. In particular, smooth communication in traded man-computer control was crucial. The results from our experimental studies indicate that when this communication is awkward, supervisory control can be inferior to direct manual control. But when the man-computer communication is efficient, the traded teleoperator control is faster and less errors result in performing a variety of complex tasks.

The use of high-level symbolic command structure has further improved performance in terms of error reduction and speed consistency. This advantage of automatic command is especially present when either the environment or the communication channel to the operator is degraded. One important conclusion of this year's study was the improvement in relative speed and operator unburdening realized through effective use of SMC^3L under degraded visibility.

Of more importance than the overall task improvement of computer control is the fact that performance is highly task dependent. Task performance of a shared control function (e.g., Resolve Motion Control and computer graphics, etc.) is inherently task specific, for it is designed to meet the specific functional requirement within a particular task element. On the other hand, performance of a traded control function (e.g., push button) or a grouped traded control (e.g., Variable or Chained commands) is less task specific, but depends on operational requirements--mainly, control specifications and transfer between task elements. Often, trade-offs in these requirements result in a mixed use of command modes: analogic and symbolic, fixed and variable rates, precanned and interpretative, etc.

The current program has resulted in the identification of potential application of SMC^3L in areas of the following characteristics:

(1) Complex task procedure with repetitive subtasks.
(2) Discrete commands with extended task time.
(3) Costly, delayed, or revealing communication.
(4) Possibility of operator overload and of subsystem autonomy.

Continued studies will identify a variety of underwater supervisory control tasks and forms of computer aiding. Although there exists diverse literature in the area of underwater task taxonomy, the determination of the relationship between task type and automation level requires a different method of classification from that which has been done in the previous work.

Each task must be categorized along a set of communication/control dimensions which differentiate between the type of automation. Among these dimensions are the task attributes identified in the current experiment studies:

(1) Control difficulty - position accuracy constraints (measured as allowance in inches); force accuracy constraints (acceptable contact force or torque).

(2) Complexity - number of active constraints (degrees-of-freedom).

(3) Variability - number and variety of objects and tools.

(4) Discreteness - level (%) decomposable to single-constraint motion steps.

(5) Uniformity - dispersion in the levels of difficulty, complexity, and variability.

(6) Repetitiveness - number and level (%) of repetition within a finite sequence of spatial transformations.

(70 Uncertainties - environmental disturbance; communication noise and feedback; machine reliability.

These variables illustrate that both constraint limits and constraint distribution are important factors. So are the uncertainties added due to environmental and machine conditions along these constraint dimensions. These variables will provide a key to selection of the appropriate communication modes.

Analysis of our findings to date leads to the following conclusions related to the design of man-machine communication language for remote manipulation.

(1) Computer-aided and manual control should be combined in a mixed initiative protocol.

(2) Task oriented commands with a sentence structure are natural to the user and easy to teach.

(3) Computer aiding can significantly improve performance but provisions should be made for manual control of fine tasks.

(4) Feedback information about the internal state of a complex computer control with complex task is essential for efficient and confident supervisory control of manipulators.

(5) The issues of feedback format and update rate are important for user acceptance. With current language design, the minimum level of feedback, which provides both input identification and display of current command, seems adequate for tasks with relatively low variability.

(6) Idea format of state feedback should be hierarchical in structure, forming spatial/visual representation. The update rate of feedback display should also be well-controlled.

The model and guidelines described here can lay the basis for a general methodology for man-machine language design. This methodology will formalize the steps necessary to transfer a task definition into a complete command system. The following are more specific steps to be taken to obtain such a methodology:

(1) Develop a task taxonomy that can help identify tasks for which a high level language is necessary.

(2) Establish the relationships between relevant task characteristics and language features that would provide the most effective communication.

(3) Establish formal language design methods within the framework of the procedural nets model:

(a) A method to formally define the language from specific task characteristics.

(b) A method to determine language primitives, hierarchy and control structures necessary from an analysis of the task structure.

(c) A method to define language structure and sequence requirements and to integrate the elements into a smooth communication scheme.

(4) Provide a standard method for evaluating the effectiveness of, and performing comparisons between, different language designs.

5. REFERENCES

Ambler, A.P., Barrow, H.G., Brown, C.M., Burstell, R.M., and Popplestone, R.J. A Versatile Computer-Controlled Assembly System. Third International Joint Conference on Artificial Intelligence, Stanford, 1973:298-307.

Bennett, J.L. The User Interface in Interactive Systems. In Annual Review of Information Science and Technology, C. A. Cuadra (ed.), Vol. 7, Washington, D.C.: American Society for Information Science, 1972:159-196.

Berson, B.L., Crooks, W.H., Shaket, E., and Weltman, G. Man-Machine Communication in Computer-Aided Manipulation. Perceptronics, Inc. (Woodland Hills, CA) Technical Report PATR-1034-77-3/1, March 1977.

Bien, A. and McDonough, P.J. Naval Applications of Man-In-The-Sea Concepts. SRI NWRC 7000-212, 1970.

Crooks, W.H., Shaket, E., and Alperovitch, Y. Man-Machine Communication in Computer-Aided Remote Manipulation. Perceptronics, Inc. (Woodland Hills, CA) March 1978.

Dijkstra, E.W. Structured Programming. Chapter 1, Academic Press, 1972.

Dijkstra, E.W. Structured Programming. Software Engineering Technology. NATO Scientific Affairs Division, Brussels 39, Belgium, 1969:84-88.

Engel, S.E. and Granada, R.E. Guidelines for Man/Display Interfaces. Poughkeepsie Laboratory, IBM Technical Report TR002720, December 1975.

Fahlman, S.E. A Planning System for Robot Construction Tasks. Artificial Intelligence, 1974, 5:1-50.

Falkoff, A. and Iverson, K.E. The Design of APL. IBM Journal of Research and Development, July 1973.

Ferrell, W.R. Command Language for Supervisory Control of Remote Manipulation. In Remotely Manned Systems, E. Heer (ed.), California Institute of Technology, 1973.

Ferrell, W.R. Delayed Force Feedback. Human Factors, October 1966.

Ferrell, W.R. Remote Manipulation with Transmission Delay. IEEE Trans. Human Factors in Electronics, 1965, HFE-6:24-32.

Ferrell, W.R. and Sheridan, T.B. Supervisory Control of Remote Manipulation. IEEE Spectrum, October 1967:81-88.

Fikes, R.E. and Nilsson, N.J. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, 1971, 2:189-208.

Finkel, R., Taylor, R., Bolles, R., Paul, R., and Feldman, J. AL, A Programming System for Automation. Stanford Artificial Intelligence Laboratory (Stanford, CA) Memo AIM-243 (STAN-CS-74-456), November 1974.

Foley, J.D. and Wallace, V.L. The Art of Natural Graphic Man-Machine Conversation. Proceedings of the IEEE, June 1975, 62(4).

Goertz, R.C. Mechanical Master-Slave Manipulator. Nucleonics, November 1954, 12:45-46.

Hill, J.W. and Sword, A.J. Studies to Design and Develop Improved Remote Manipulator Systems. Stanford Research Institute, NASA CR 2238, April 1973.

Iverson, K.E. A Programming Language. New York: Wiley, 1962.

Johnsen, E.G. and Corliss, W.R. Teleoperators and Human Augmentation. NASA SP-5047, 1967.

Kennedy, T.C.S. The Design of Interactive Procedures for Man-Machine Communication. International Journal of Man-Machine Studies. 1974, 6:309-334.

Lozamo Perez, T. and Winston, P.H. LAMA: A Language for Automatic Mechanical Assembly. (Boston, MA) Fifth International Joint Conference on Artificial Intelligence. August 1977:710-716.

Nevin, J.L., Sheridan, T.B., Whitney, D.E. and Woodin, A.E. The Multi-Moded Remote Manipulator System. In Proceedings First National Conference on Remotely Manned Systems, E. Heer (ed.) California Institute of Technology, 1973.

Nevin, J.L. and Whitney, D.C. Computer Controlled Assembly. Scientific American, January 1978.

Ocean System, Inc. What Do Those Expensive Divers Really Do? Offshore Services Magazine, September 1977:78-81.

Pesch, A.J., Hill, R.G., and Allen, F.L. At-Sea Operator Performance of Small Submersible Manipulators. General Dynamics Electronic Boat Division Technical Report No. U-413-71-031, 1971.

Pesch, A.J., Klepser, W.F., Hill, B.G., and Simoneau, G.R. Operator Performance with Alternate Forms of Unilateral Position Control for Undersea Manipulators. General Dynamics Electric Boat Division, Report No. U413-72-051, September 1972.

Pesch, A.J., Hill, B.G., and Klepser, W.F. Capabilities of Operators as Divers: Submersible Manipulator Controllers in Undersea Tasks. General Dynamics Corporation (Groton, CT) AD-716532, June 1970.

Sacerdoti, E.D. Planning in a Hierarchy of Abstraction Spaces. Third International Joint Conference on AI, 1973:412-422.

Sacerdoti, E.D. A Structure for Plans and Behavior. Stanford Research Institute Artificial Intelligence Center, Technical Note 109, 1975.

Schneider, M.H. Task Analysis for Undersea Manipulators. M.S. Thesis, M.I.T., 1977.

Sheridan, T.B. and Verplank, W.L. Human and Computer Control of Undersea Teleoperators. Man-Machine System Laboratory, M.I.T., 1978.

Treu, S. Interactive Command Language Design Based on Required Mental Work. International Journal of Man-Machine Studies. 1975, 7:135-149.

Verplank, W.L. Symbolic and Analogic Command Hardware for Computer-Aided Manipulation. M.S. Thesis, M.I.T., 1967.

Von Neumann, J. Collected Works 5. (Written 1946), New York: Macmillan, 1963.

Wang, S.J. A Programmable Industrial Robot Control System. IEEE Transactions on Systems, Man and Cybernetics, 1976, SMC-6(8):570-580.

APPENDIX A

COMPUTER CONTROLLED MANIPULATION FACILITY

COMPUTER-AIDED MANIPULATION FACILITY

1. Overview

Perceptronics' computer-aided manipulator facility includes a hydraulic servo manipulator, a minicomputer, and a man-machine interface (Figure A-1). An operator controls the manipulator through the joysticks and pushbottoms of the control console; he observes the manipulator activities through the two-view TV displays. The operator's inputs are processed by the minicomputer; the minicomputer, in turn, controls the servo manipulator and responds to the manipulator's position-sensing potentiometers. Data communications between the minicomputer and the control console or manipulator electronics occur via the programmable interface. A review of the command language design is given in Chapter 2. The individual components of the manipulator and interface are described in the following sections.

2. Servoarm Manipulator

The Servoarm manipulator, shown in Figure A-2 is electronically-controlled and hydraulically-powered. The manipulator has six rotating joints (each with a full 180^{o} movement range) plus gripper closure. The arm motions and joint numbers are (in anthropomorphic notation):

(1) Shoulder retention.
(2) Shoulder elevation.
(3) Elbow flexion.
(4) Forearm rotation.
(5) Wrist flexion.
(6) Gripper rotation.

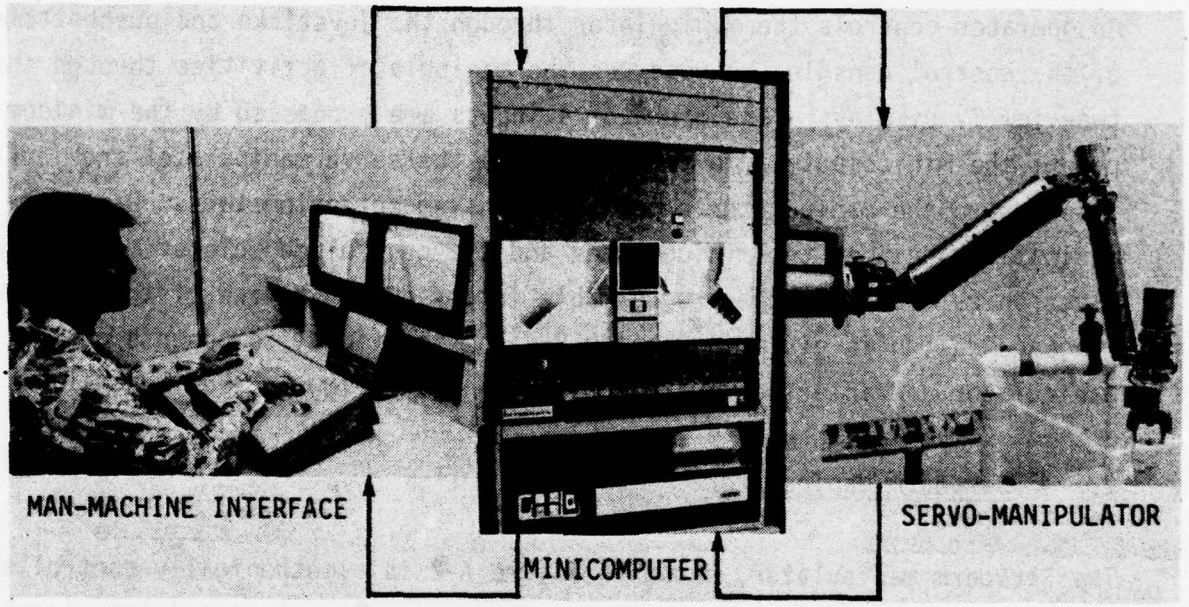


FIGURE A-1.
COMPUTER AIDED MANIPULATOR FACILITY

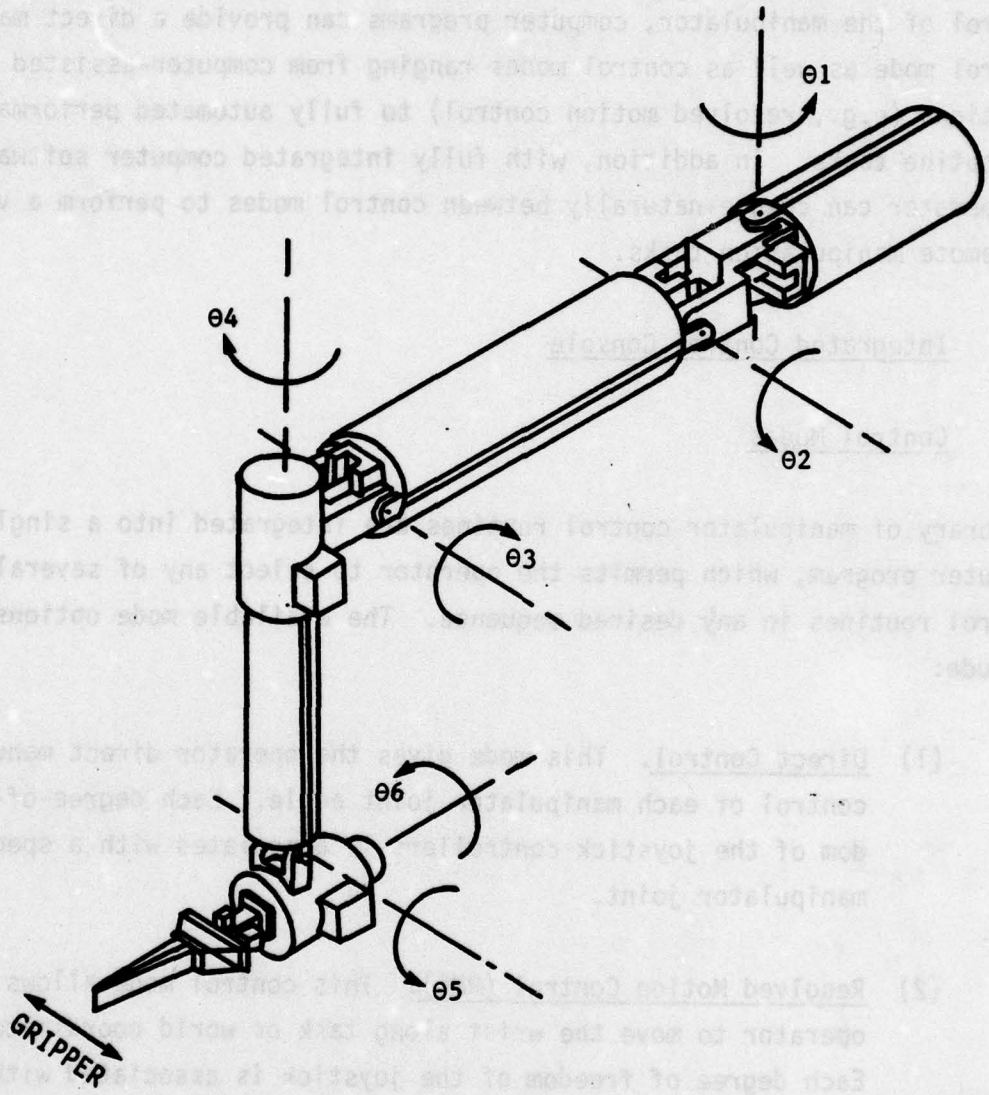


FIGURE A-2.
SERVO MANIPULATOR WITH MOTIONS
OF THE SIX ROTARY JOINTS

These motions provide the six degrees of freedom necessary to position and orient an object in the work space. With the computer in direct control of the manipulator, computer programs can provide a direct manual control mode as well as control modes ranging from computer-assisted functions (e.g., resolved motion control) to fully automated performance of routine tasks. In addition, with fully integrated computer software, an operator can change naturally between control modes to perform a variety of remote manipulation tasks.

3. Integrated Control Console

3.1 Control Modes

A library of manipulator control routines are integrated into a single computer program, which permits the operator to select any of several control routines in any desired sequence. The available mode options include:

(1) Direct Control. This mode gives the operator direct manual control of each manipulator joint angle. Each degree-of-freedom of the joystick controllers is associated with a specific manipulator joint.

(2) Resolved Motion Control (RMC). This control mode allows the operator to move the wrist along task or world coordinates. Each degree of freedom of the joystick is associated with movement of the manipulator end-effector along a specific X, Y, or Z axis of the work space. The operator specifies the speed and direction of motion of the manipulator wrist, and the computer calculates the required angle of each joint and outputs these as commands to the individual joints of

the manipulator. RMC frees the operator from the responsibility of determining which combination of speed and motions will produce the required trajectory.

(3) Automatic Commands. Under Automatic Motion Control (AMC), the computer assumes control and moves the arm from its current location to any preassigned location. Under Fixed Commands, the computer performs single unitary motions through the operator's keyboard commands. Under Variable command, the computer records and moves the arm automatically to previous recorded configurations (extended AMC); accepts task procedure and supervises automatic execution of complex tasks, with possible intervention and modification by the operator. Under Chained command, the computer decodes task procedures and performs the functions under the Variable Command mode.

(4) Speed Adjustment. This option allows the operator to select the maximum rate of manipulator motion. The joysticks are rate controllers with greater stick deflection, providing faster manipulator movement. The speed adjustment routine allows the operator to select one of the three rates over which the joysticks function.

(5) Other System Features. This includes enable/disable of wrist invariance function, enable/disable of force sensor messages, and other command feedback features, etc.

3.2 Console Configuration

The man-machine interface consists of a control console by which an operator can manually control arm motions, select computer assistance control functions, and observe control status.

The operator uses both joysticks and the pushbuttons to control the manipulator. The joysticks are used for manual control, and the pushbuttons are used for control mode selection. Using the pushbuttons and joysticks, the operator can smoothly take the manipulator through a sequence of tasks, selecting control modes, rates, and manual operations that are most appropriate for each subtask. For example, he may use Automatic Command for gross movement from the stowed position to the target area. He can then change immediately to RMC for fine movements and for alignment and insertion. While in RMC mode, the operator can move the arm to the "drop" point, and then record the latter point to facilitate repetition of the task.

Three CRT displays are used to provide system state feedback and two-view TV viewing of the manipulator work space. The details are discussed in Chapter 3.

4. Computer Control System.

4.1 Central Processor.

The supporting processor for the manipulator system is an Interdata Model 70 minicomputer. In includes 48 k-bytes of core memory, a high speed paper tape reader/punch, a disk memory and a re-settable precision interval clock. The disk drive, CRT, and other peripherals are used to support program development work and are not part of the real time manipulator control system.

4.2 Processor Interface.

Data transfer between the computer and manipulator servo-electronics and between the computer and the control console is performed by a Perceptronics +I/O Programmable Interface.

The +I/O Interface contains a number of functional modules arranged along a transfer buss by which commands, data and status signals are communicated. These modules perform such individual functions as standardizing communication with the processor, sequencing data transfers across the buss, and performing D/A conversion and output.

In addition to an interface module between the processor and interface, the +I/O includes an A/D module that is used to interface with the servo position potentiometers and control console joysticks. Thirty-two individually addressable input channels are provided. An eight-channel D/A module is used for converting and sending position commands (voltages) to the control inputs of the servos. Finally, a D/D module is used to provide the 16 input and output channels for the button and lamp arrays of the control console.

4.3 Software

The software system includes three main modules shown in Figure A-3. Module 1 is the main control process. It contains the central loop which executes most of the software functions. All the programs of primitive and non-primitive functions, joystick control and state command programs are included in this module. Module 2 is the teletype process module. It contains the programs which read the keyboard codes, then analyzes and interprets them. Module 3 is the I/O process module. It contains programs which "read" and "write" and send characters to the CRT. A detailed software documentation for new and modified functions in the current year is presented in Appendix B.

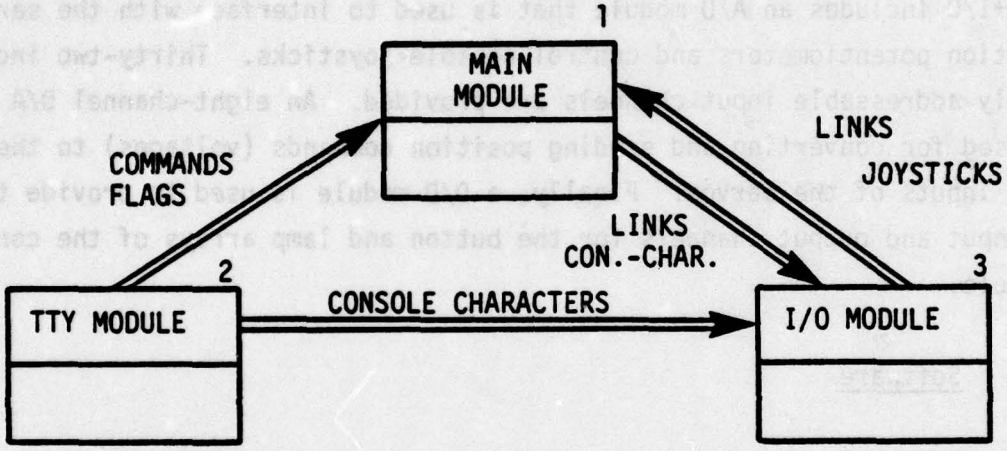


FIGURE A-3.
DATA CONNECTION DIAGRAM

APPENDIX B

NEW SYSTEM SOFTWARE DOCUMENTATION

TABLE OF CONTENTS

LIST OF

CHARTS, TABLES, & FIGURES

1. INTRODUCTION

1.1 Old System Functional Overview

The ONR-1977 Computer-Aided Manipulator Facility includes a hydraulic servo manipulator, man-machine interface (controls & console) and a minicomputer. The operator controls the manipulator through joysticks and pushbuttons on the control & console and observes the manipulator's activities through the two-view TV displays.

Three control modes were available:

(1) Direct joints control.

(2) Resolved motion control.

(3) Automatic motion control.

The command language is based on the procedural net model analysis of command language requirements and the principles identified by the analysis and by review of related work. This language consists of a set of primitive commands and "chains" or sequence commands. A detailed software documentation was included in last year's technical report: Man-Machine Communication in Computer-Aided Remote Manipulation, PATR-1034-78-3, Perceptronics, Inc., March, 1978.

1.2 New System Functional Description

Perceptronics' 1978 task of manipulator's software was developed in two stages:

(1) Continuing development and incorporation of functions that were designed during 1978.

(2) Modification and enhancement of operating functions resulted from the intensive experimentation of last year.

To enhance the communication language function, the "embedding-a-chain-within-a-chain" capability was implemented. When such nesting was allowed, the hierarchical planning and execution of a task could be performed by the operator. A wrist force sensor unit was also incorporated within the system. The sensor software provided the operator two types of force warning and stop messages.

Control Feedback Display. The control feedback display subsystem was modified and enhanced. Selective feedback display can be chosen using panel switches. Queue display was changed from the swiftly changing frames used last year into a static format which uses dynamic pointer. Feedback display of chain execution was added.

Automatic Monitoring. An automatic monitoring function was added into the system. This subsystem includes two components.

(1) On-line monitoring of events occurring in the manipulator's system.

(2) Off-line recording module which copies monitored data from core into a file and generates events sequence reports.

2. SYSTEMS' NEW AND MODIFIED FUNCTIONS

2.1 Selective Feedback Display

Before starting an experimental session the experimenter has the option to enable/disable, partially or completely, the feedback display system. The following combinations of feedback state are possible:

(1) Full display: where queue and chain execution is displayed as well as chain definition.

(2) Partial display: where queue and chain execution is displayed, while chain definition is not.

(3) Partial display: where chain definition is displayed and queue and chain execution is not.

(4) No display: where queue and chain execution and chain definition are not displayed.

Each of the four display modes is selected by a unique setting of a single key on the front panel of the computer, shown in Table 2-1.

When the right switch is set and a session starts, it is impossible to change the mode of display. To change it, a session must be terminated by resetting all switches on the panel, then setting a different switch on the panel and restarting the session over again. If the illegal switches combination is set, the system assumes a default mode of full display.

TABLE 2-1

FEEDBACK DISPLAY MODE SELECTION

PANEL SWITCH NO. / DISPLAY MODE	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15
1	| o o o	o o o o	o o o o	o o o o
2	o o o o	| o o o	o o o o	o o o o
3	o o o o	o o o o	| o o o	o o o o
4	o o o o	o o o o	o o o o	| o o o

2.2 Queue Feedback Display

The ONR77 system, displayed commands of the queue in a pipe-like fashion. Every time the command execution was finished or a new command was entered at the keyboard, the whole queue was erased and redisplayed automatically in its new state. This form of feedback, though it reflected accurately the state of the queue, overburdened the operator and the CRT display mechanism, especially at times of frequent feedback events in the system.

The need for a more stable feedback display, which should reflect a highly dynamic concurrent system, yielded a new feedback mechanism concept. Two queueing mechanisms were constructed. The first reflects the command execution process in the system and is displayed on the CRT. The second is an invisible queue, which contains commands typed in at the keyboard. The system executes commands of the visible queue.

The queue is statically displayed on the screen, and the command at execution is indicated by a pointer that moves down the queue to the next command when execution of the current command terminates. The only dynamic part is the pointer, which moves down the queue as command execution propagates. This new form of display enables much easier and quicker perception of the dynamic changes occurring in the sequence of command execution.

While the system executes commands in the visible queue the operator can enter new commands at the keyboard. These commands are stored into the invisible queue which can contain up to ten commands. When execution of the last command in the visible queue terminates, it is erased from the screen, and the content of the invisible queue is copied into the

visible one. New content of the visible queue is then displayed on the screen and command execution resumes until the cycle repeats itself.

This new configuration handles independently two concurrent processes. The visible queue is redisplayed every time these two processes interact at a much lower frequency than in the previous system. Overall, the double queue mechanism is an effective technique to reflect a highly dynamic process on the background of a static domain.

The system can contain, at most, twenty commands, ten of which can be in the visible queue and ten in the invisible queue. Any additional commands entered when the invisible queue is filled are lost.

2.3 Nested Feedback Display (Chain Calls)

An additional feature, which utilizes the concept of static domain display, is the first level nesting feedback display. A chain called from the queue is performed at the first level nesting. This chain can then call another chain, to create a nesting of up to five levels. (Recursion is prohibited and automatically aborted.) The first, of five possible nesting levels, is displayed on the screen when being executed. While the pointer of the visible queue points to the chain call, the chain is expanded at the center of the screen and another pointer indicates execution of the chain's commands.

Such a situation of call to chain 1 is displayed in Figure 2-1. If chain 1 would call chain 2 (second level nesting) the pointer of chain 1 would point to chain 2 call, and the execution of chain 2 would be performed invisibly. When control returns to chain 1, its pointer moves downward until the chain terminates and is erased from the screen.

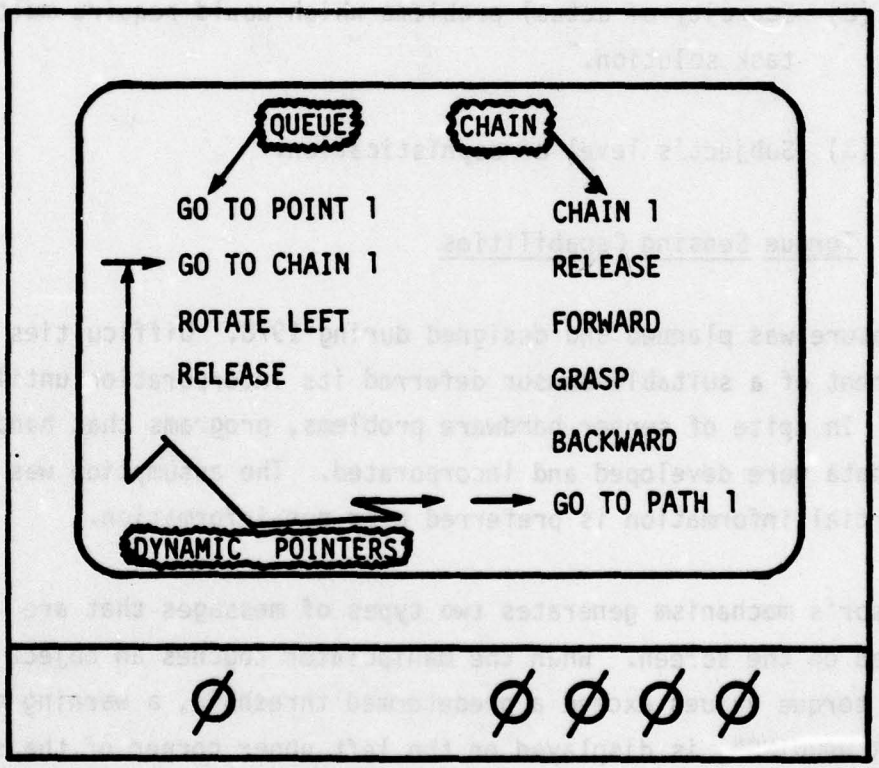


FIGURE 2-1.
CRT QUEUE AND CHAIN EXECUTION DISPLAY

Then, the pointer of the queue resumes control moving downward, and so on.

The system does not support visual feedback of the second through the fifth levels of nesting because of:

(1) Limited display space on the screen.

(2) Scarcity of actual problems which would require multi-level task solution.

(3) Subject's level of sophistication.

2.4 Torque Sensing Capabilities

This feature was planned and designed during 1978. Difficulties in the development of a suitable sensor deferred its incorporation until recently. In spite of sensor hardware problems, programs that handle the sensor data were developed and incorporated. The assumption was that even partial information is preferred over non-information.

The sensor's mechanism generates two types of messages that are displayed on the screen. When the manipulator touches an object and the applied torque values exceed a predetermed threshold, a warning message ("SENSOR WARNING") is displayed on the left upper corner of the screen. When this condition terminates, the message disappears. On the other hand, when the torque continues to build up and input values are exceeded to a second predetermined threshold, a sensor stop message, ("SENSOR STOP") is displayed on the screen and a bell rings. At the same time, the global stop flag, which disables all automatic command execution, is set and the system waits for operator's manual intervention.

After the operator moves the arm away from the obstacle, through joystick operation, he enables the automatic command execution by pushing the ("CONTINUE") button which also erases the sensor stop message from the screen.

When the sensor is continuously in contact with an object, the warning message stays on the screen. The operator has a choice to disregard it and continue with the automatic or manual task. The warning message is useful when a hidden side of the manipulator touches an object. In case of excessive false alarms, the operator can disable the sensor mechanism.

2.5 Data Monitoring System

2.5.1 Overview. This system consists of two different and physically separate parts:

(1) Real time data acquisition and monitoring.

(2) Off-line data recording and report generating.

While the first function is incorporated into the ONR78 software package, the second one is a stand alone module to be executed after each experimental session.

2.5.2 Real Time Monitoring. Last year's experiments showed that in order to perform a detailed data analysis, an automatic on line data collection system is required. In particular, the high frequency of event occurrences in the manipulator control environment has made precise manual recording impractical. Basically, four types of event

occurrence and their corresponding time are recorded within the present design:

(1) Keyboard input commands.

(2) Commands in execution.

(3) Time intervals when joystick is operated.

(4) Special syntax errors.

These data are generated by different system processes; therefore, they must be acquired at different software locations. For example, keyboard input commands and special events are intercepted by routines in the teletype process, while commands in execution and joystick operation intervals are handled by routines in the main process.

The real time system constraints do not allow direct file recording and therefore, the acquired data are stored in the core. Available core permits monitoring of 60 to 90 minutes of experimentation, depending on the intensity of the session. During experimentation, the current core monitoring address is shown on the front panel, indicating available free core.

2.5.3 Data Recording and Report Generation. This is an off-line stand alone package which is activated immediately after session termination. The monitored data which reside in core are copied into a file and a report of events history is printed out. An example of the report is shown in Appendix I.

The monitored data starts at a given address in core and its first address contains the end address of monitored data. The package writes the core content in a sequential form into a disc file. Afterwards, it consecutively decodes events' time and type and prints them out.

Appendix II shows the hexadecimal codes as they are recorded in core and their meaning. The printed report is a sequential reconstruction of events on a time scale. Analysis of parallel processes (such as execution and command entry) requires further analysis.

3. SOFTWARE DOCUMENTATION

3.1 Software Hierarchical and Functional Structure

The modified hierarchical structures of the main process and the teletype process are similar to last year's report. The I/O process was not modified. Only the new or modified programs in the main process are discussed and some of them are presented in flowchart diagrams.

3.2 Main Process

The modified hierarchical structure of the main process is shown in Figure 3-1.

The following list describes modified and newly developed programs which were incorporated into the main process module:

EXPER - has quite a few small modifications introduced into its program in order to accommodate changes in the system. The main modification was introduced into the primitives' management section, which was actually shortened and includes a call to commands-in-execution monitoring program. This section is shown in Chart 1.

INTAB - was changed to initialize only two paths instead of five and to invoke sensor calibration routine.

READSP - is a new program which set the feedback display system as described in Chapter 2.1. It is shown in Chart 2.

INTRP - was significantly modified to accommodate the new mechanism of split queue. Its new flowchart is shown in Chart 3.

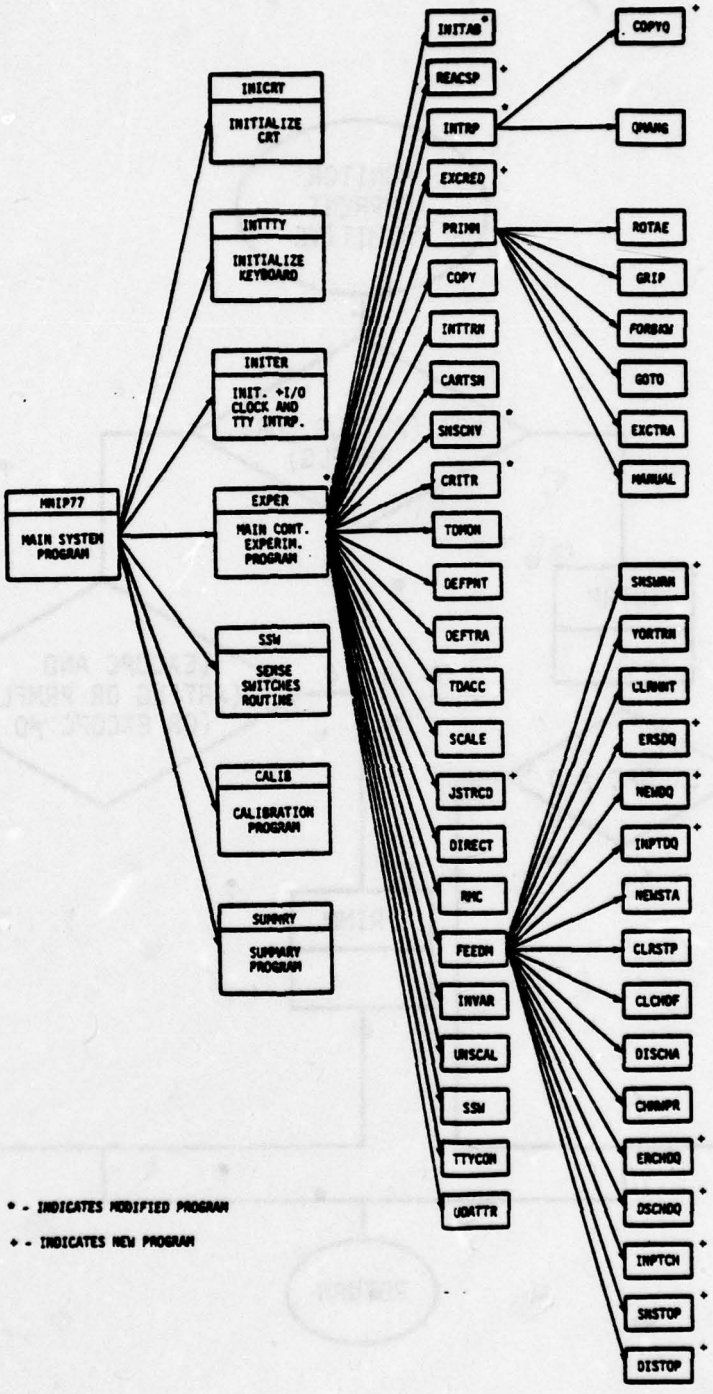


FIGURE 3-1.
MAIN MODULE HIERARCHICAL TIER STRUCTURE

CHART 1

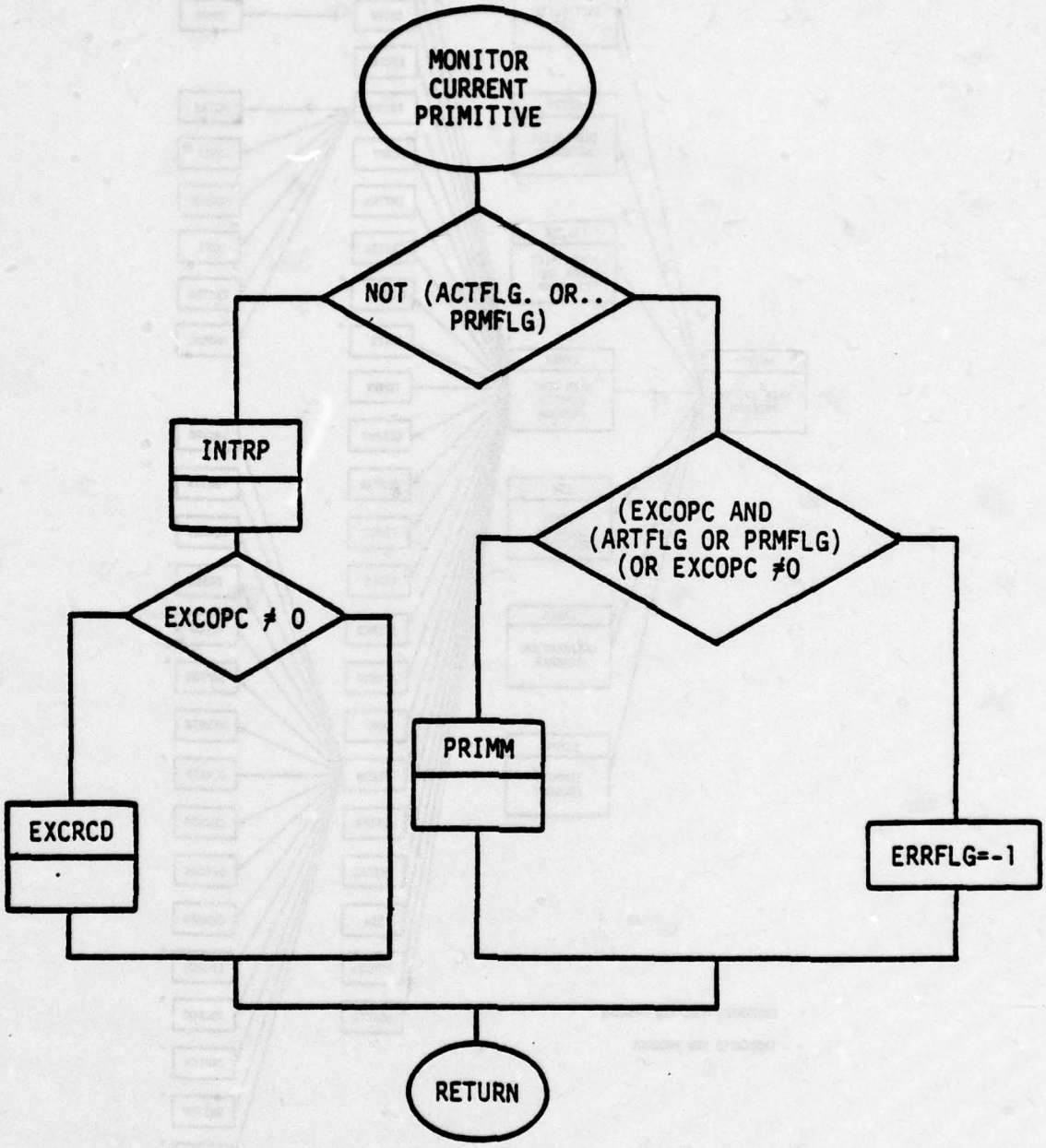

MONITOR CURRENT PRIMITIVE
NOT (ACTFLG. OR.. PRMFLG)
INTRP
EXCOPC ≠ 0
EXCRCD
(EXCOPC AND (ARTFLG OR PRMFLG) (OR EXCOPC ≠0
PRIMM
ERRFLG=-1
RETURN

CHART 2

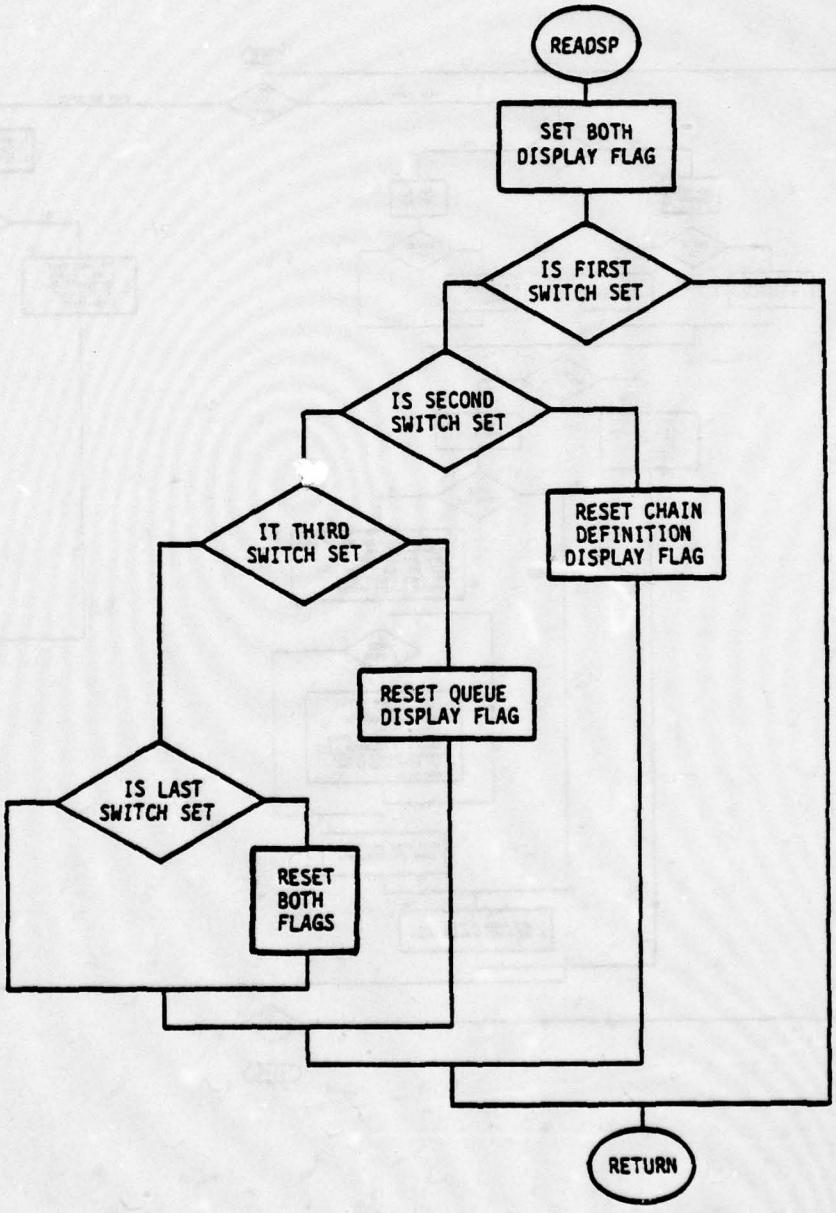
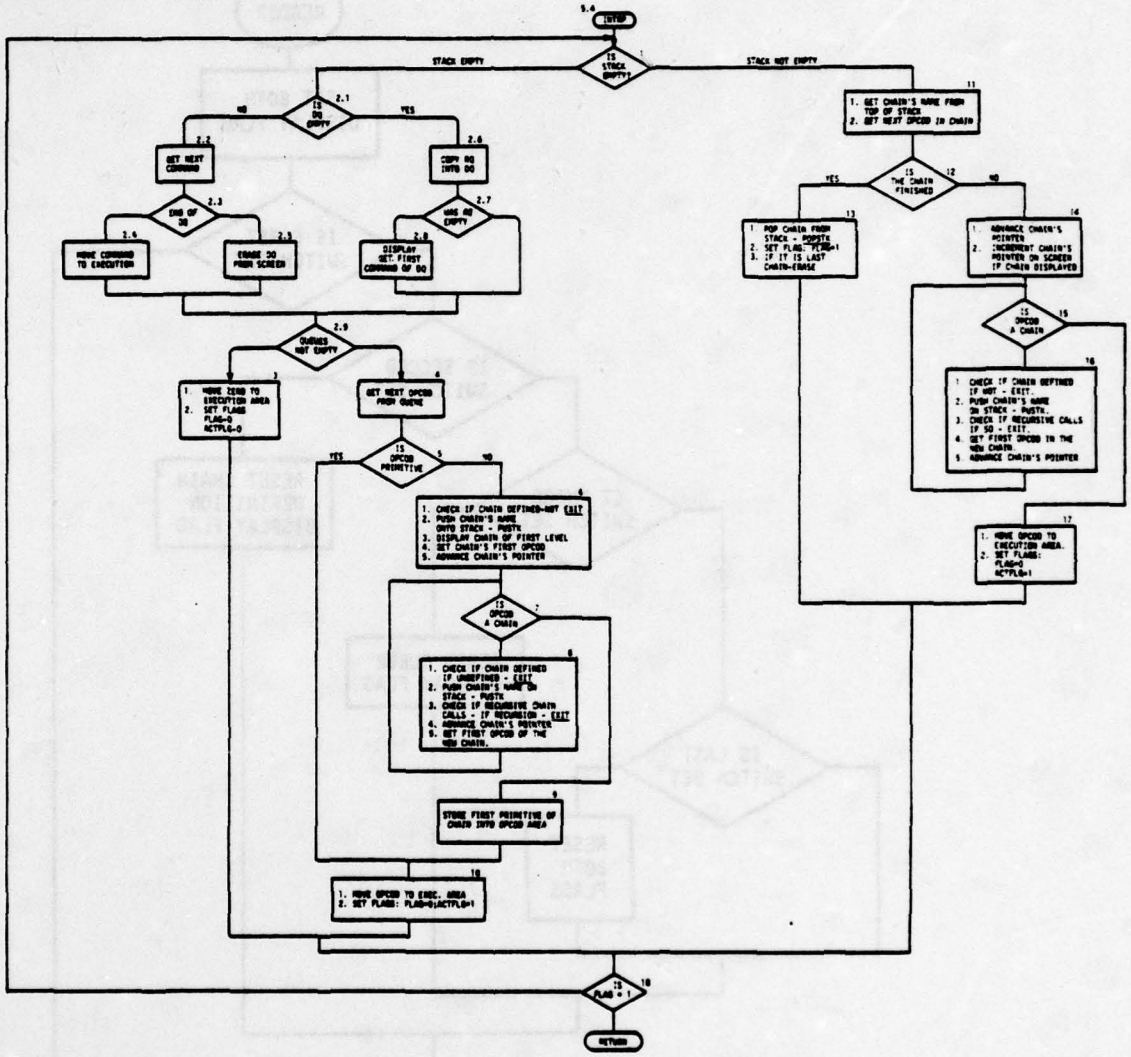

READSP
SET BOTH DISPLAY FLAG
IS FIRST SWITCH SET
IS SECOND SWITCH SET
IT THIRD SWITCH SET
RESET CHAIN DEFINITION DISPLAY FLAG
RESET QUEUE DISPLAY FLAG
IS LAST SWITCH SET
RESET BOTH FLAGS
RETURN

CHART 3



EXCRED - is invoked to monitor commands which are submitted to execution. It is a part of the real time monitoring system.

SNSCNV - was rewritten completely in order to extract maximum information from the faulty sensor. It processes the raw input data coming from the sensor and checks for threshold excess values. It is shown in Chart 4.

CRITER - was also rewritten to generate indication of warning and stop conditions due to sensor inputs. It is shown in Chart 5.

JSTRCD - is the second real time monitoring program, which monitors joystick activity whenever it is operated.

FEEDM - calls additional routines now to display sensor stop and warning and the visual feedback routines of the new queue display mechanism.

COPYQ - copies the invisible (RQ) into the visible queue (DQ) and is shown in Chart 6.

NEWDQ - displays on screen a new visible queue as shown in chart 7.

DSCHDQ - displays on screen a chain in execution as shown in chart 8.

INPTDQ - increments pointer of displayed chain as shown in Chart 9.

ERASDQ - erases from screen displayed queue as shown in Chart 10.

INPTCH - increments pointer of displayed chain as shown in Chart 11.

CHART 4

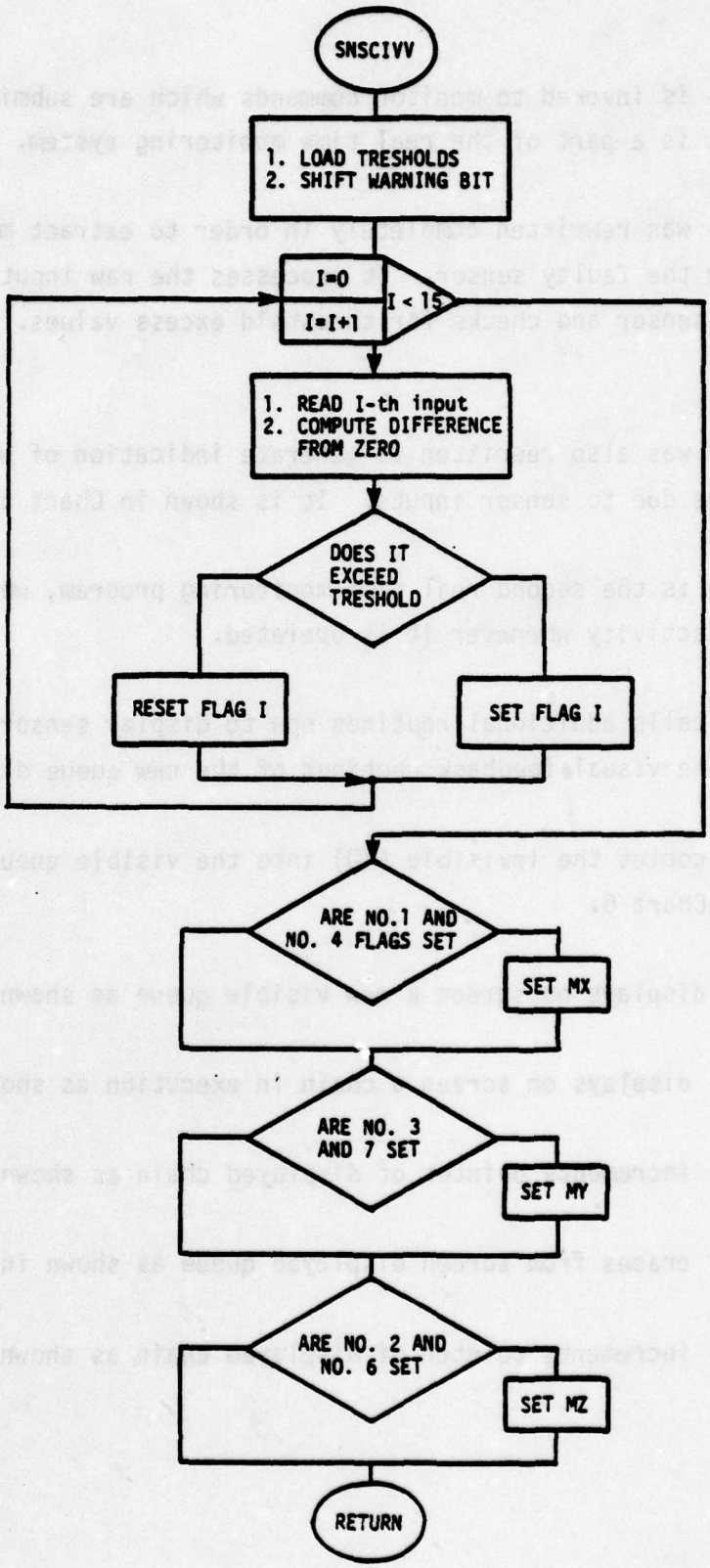

SNSCIVV
1. LOAD TRESHOLDS
2. SHIFT WARNING BIT
I=0
I=I+1
I < 15
1. READ I-th input
2. COMPUTE DIFFERENCE FROM ZERO
DOES IT EXCEED TRESHOLD
RESET FLAG I
SET FLAG I
ARE NO.1 AND NO. 4 FLAGS SET
SET MX
ARE NO. 3 AND 7 SET
SET MY
ARE NO. 2 AND NO. 6 SET
SET MZ
RETURN

CHART 5

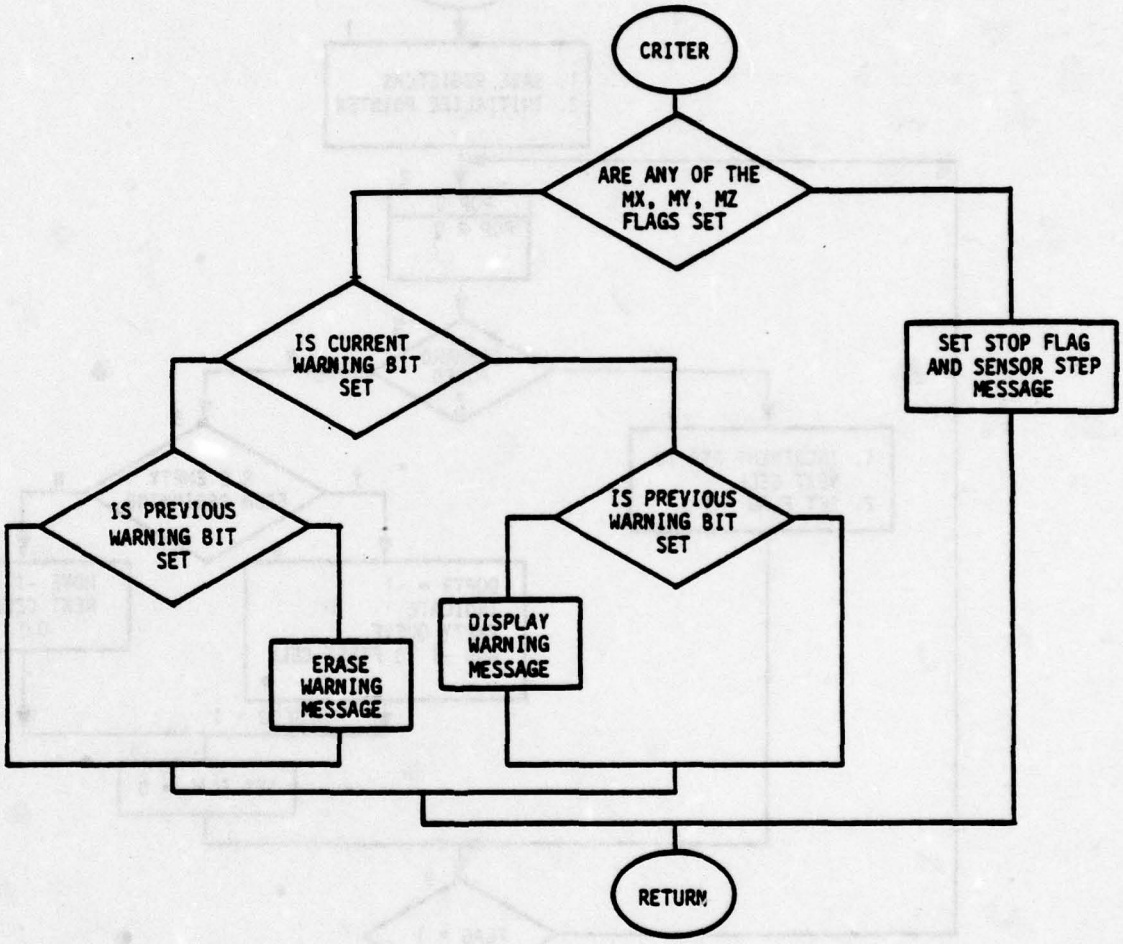

CRITER
ARE ANY OF THE MX, MY, MZ FLAGS SET
IS CURRENT WARNING BIT SET
SET STOP FLAG AND SENSOR STEP MESSAGE
IS PREVIOUS WARNING BIT SET
IS PREVIOUS WARNING BIT SET
ERASE WARNING MESSAGE
DISPLAY WARNING MESSAGE
RETURN

CHART 6

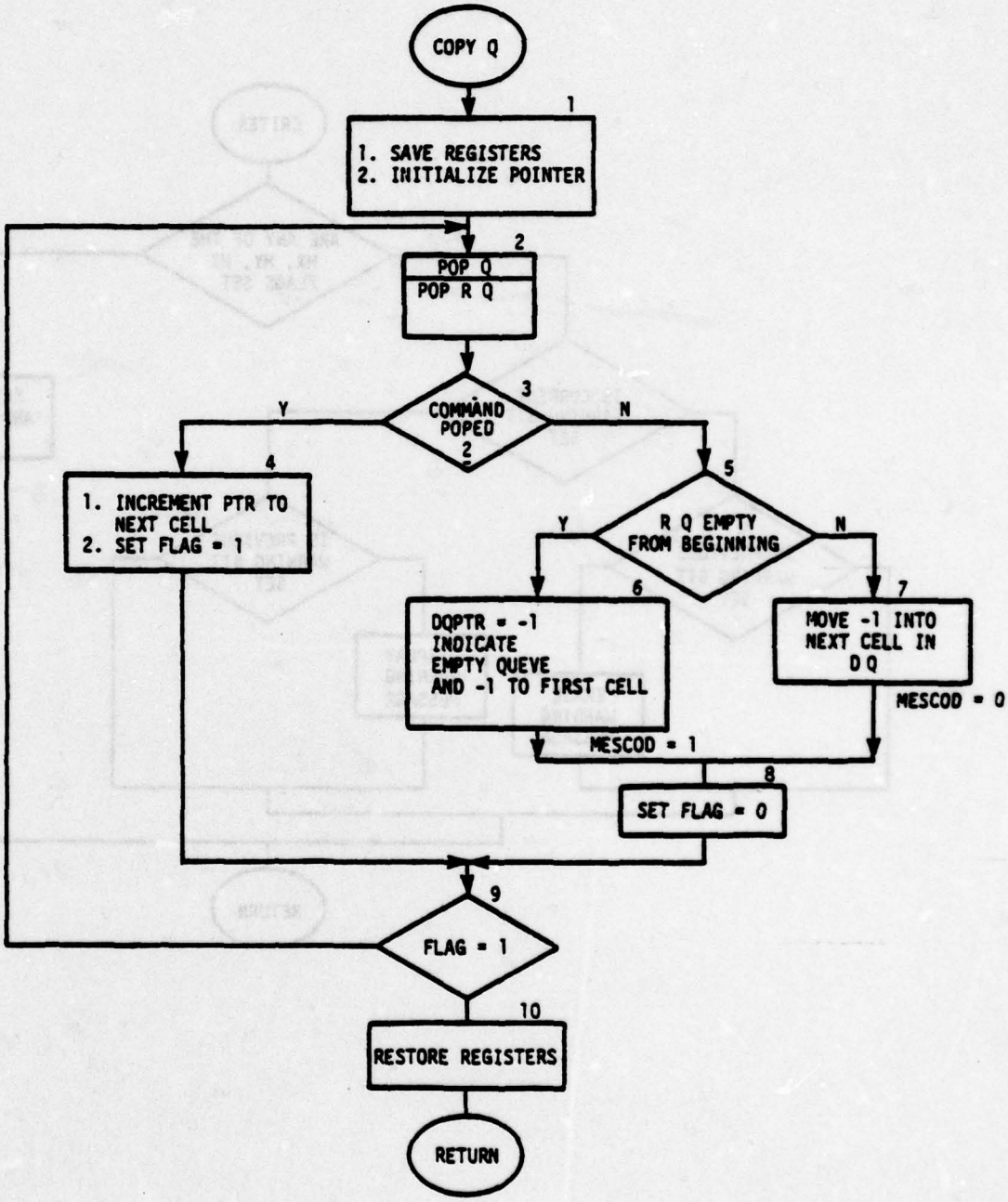

COPY Q
1
1. SAVE REGISTERS
2. INITIALIZE POINTER
2
POP Q
POP R Q
3
COMMAND POPED 2
Y
N
4
1. INCREMENT PTR TO NEXT CELL
2. SET FLAG = 1
5
R Q EMPTY FROM BEGINNING
Y
N
6
DQPTR = -1
INDICATE EMPTY QUEVE
AND -1 TO FIRST CELL
7
MOVE -1 INTO NEXT CELL IN D Q
MESCOD = 0
MESCOD = 1
8
SET FLAG = 0
9
FLAG = 1
10
RESTORE REGISTERS
RETURN

CHART 7

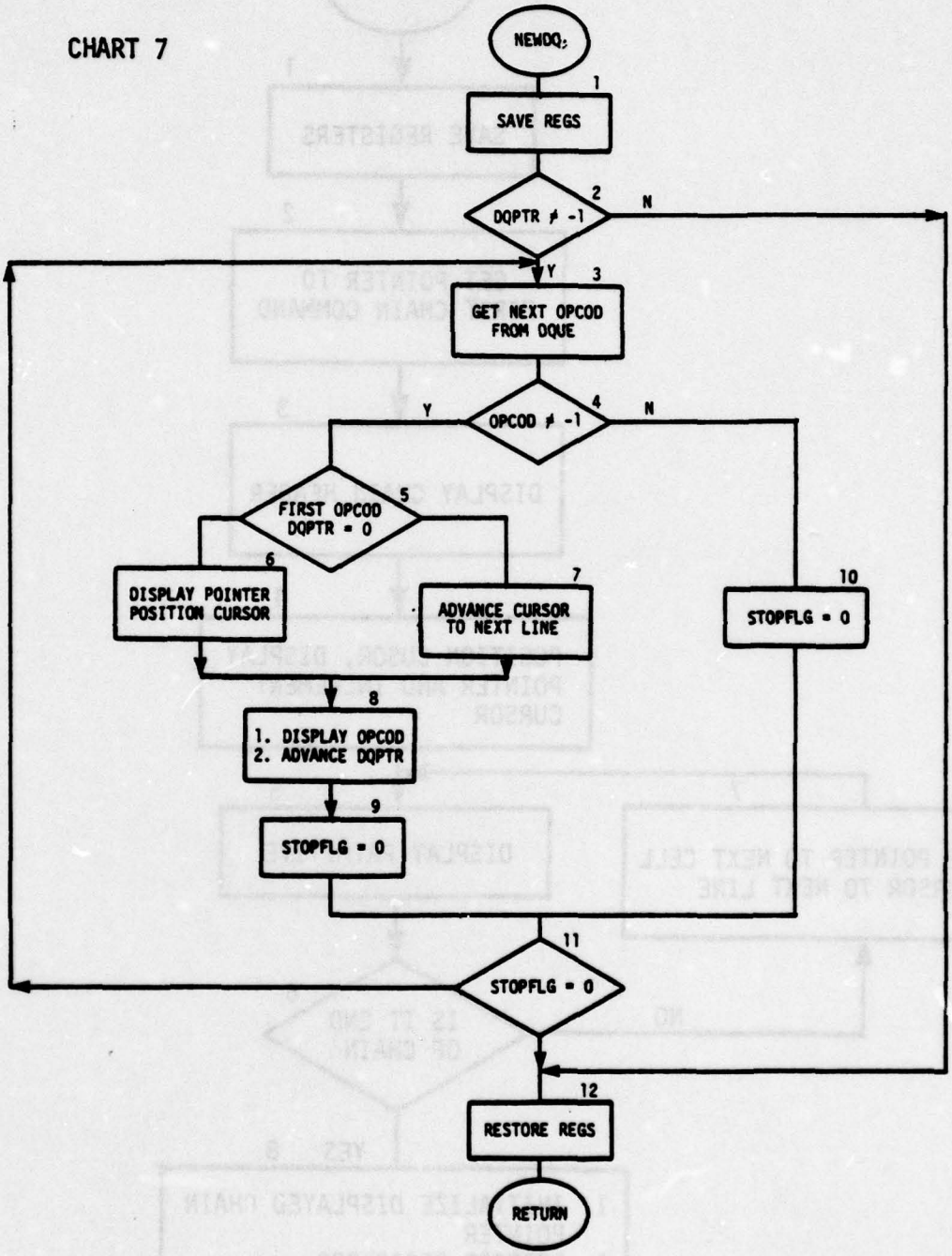

NEWDQ:
1
SAVE REGS
2
DQPTR ≠ -1
N
Y
3
GET NEXT OPCOD FROM DQUE
Y
4
OPCOD ≠ -1
N
5
FIRST OPCOD DQPTR = 0
6
DISPLAY POINTER POSITION CURSOR
7
ADVANCE CURSOR TO NEXT LINE
10
STOPFLG = 0
8
1. DISPLAY OPCOD
2. ADVANCE DQPTR
9
STOPFLG = 0
11
STOPFLG = 0
12
RESTORE REGS
RETURN

CHART 8

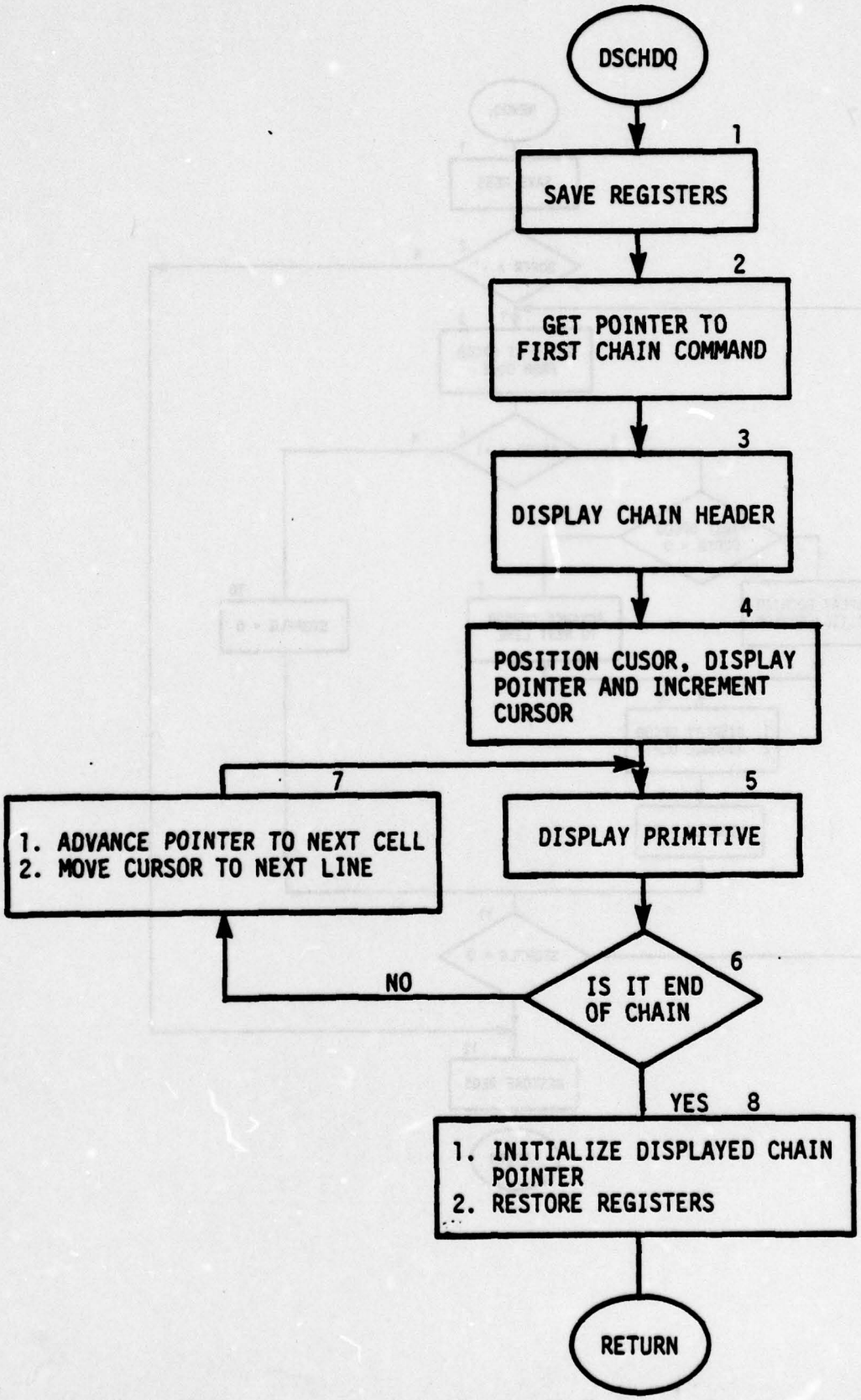

DSCHDQ
1
SAVE REGISTERS
2
GET POINTER TO FIRST CHAIN COMMAND
3
DISPLAY CHAIN HEADER
4
POSITION CUSOR, DISPLAY POINTER AND INCREMENT CURSOR
7
1. ADVANCE POINTER TO NEXT CELL
2. MOVE CURSOR TO NEXT LINE
5
DISPLAY PRIMITIVE
6
IS IT END OF CHAIN
NO
YES
8
1. INITIALIZE DISPLAYED CHAIN POINTER
2. RESTORE REGISTERS
RETURN

CHART 9

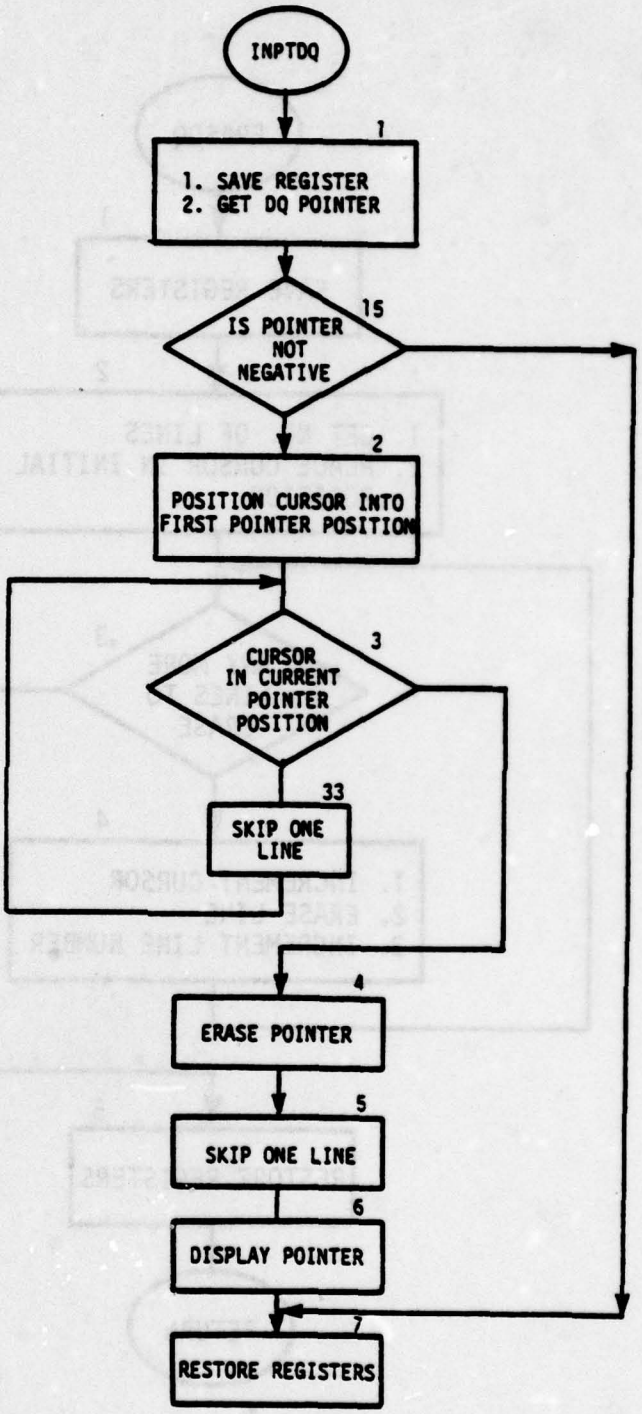

INPTDQ
1
1. SAVE REGISTER
2. GET DQ POINTER
15
IS POINTER NOT NEGATIVE
2
POSITION CURSOR INTO FIRST POINTER POSITION
3
CURSOR IN CURRENT POINTER POSITION
33
SKIP ONE LINE
4
ERASE POINTER
5
SKIP ONE LINE
6
DISPLAY POINTER
7
RESTORE REGISTERS

CHART 10

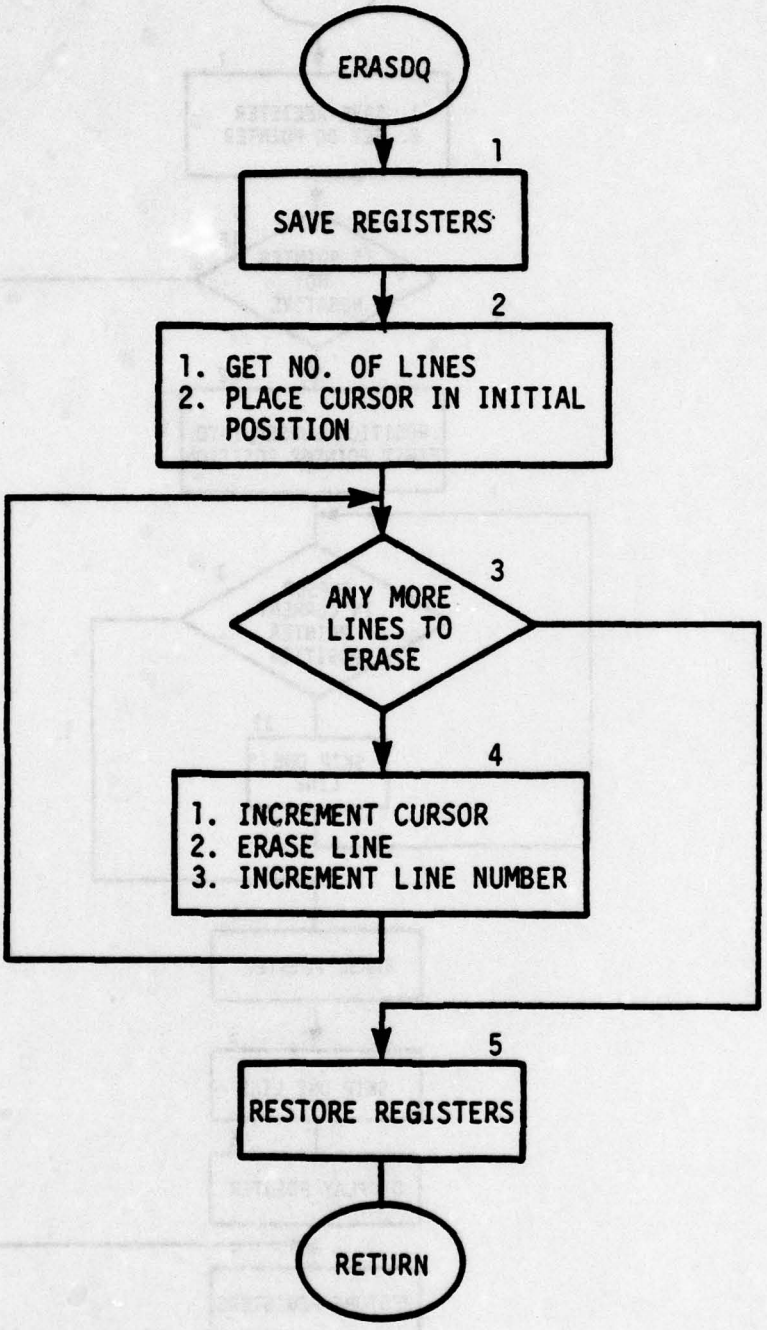
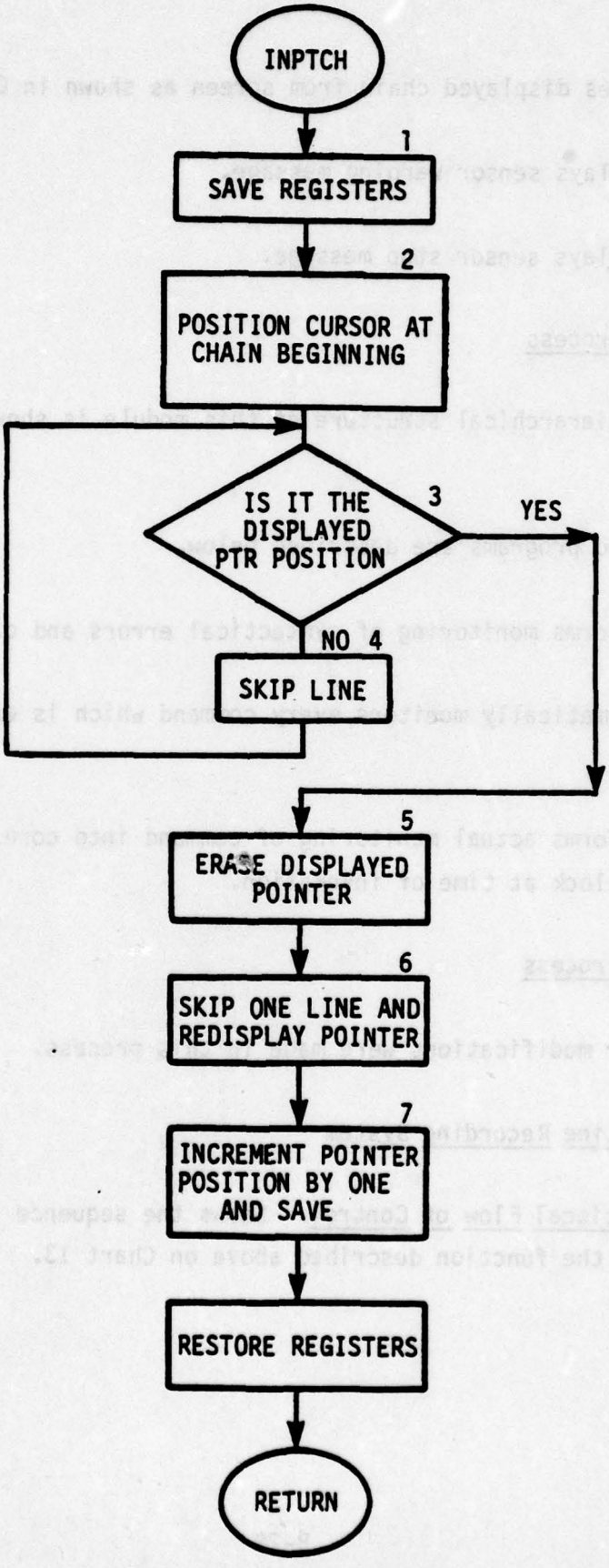

ERASDQ
1
SAVE REGISTERS
2
1. GET NO. OF LINES
2. PLACE CURSOR IN INITIAL POSITION
3
ANY MORE LINES TO ERASE
4
1. INCREMENT CURSOR
2. ERASE LINE
3. INCREMENT LINE NUMBER
5
RESTORE REGISTERS
RETURN

CHART 11


INPTCH
1
SAVE REGISTERS
2
POSITION CURSOR AT CHAIN BEGINNING
3
IS IT THE DISPLAYED PTR POSITION
YES
NO 4
SKIP LINE
5
ERASE DISPLAYED POINTER
6
SKIP ONE LINE AND REDISPLAY POINTER
7
INCREMENT POINTER POSITION BY ONE AND SAVE
RESTORE REGISTERS
RETURN

ERCHDQ - erases displayed chain from screen as shown in Chart 12.

SNSWRN - displays sensor warning message.

SNSTOP - displays sensor stop message.

3.3 TTY Process

The modified hierarchical structure of this module is shown in Figure 3-2.

New or modified programs are described below.

ANLIZR - performs monitoring of syntactical errors and cancelations.

KBDRCD - automatically monitors every command which is entered on the keyboard.

PRFRCD - performs actual monitoring of command into core prefixed by value of the clock at time of invokation.

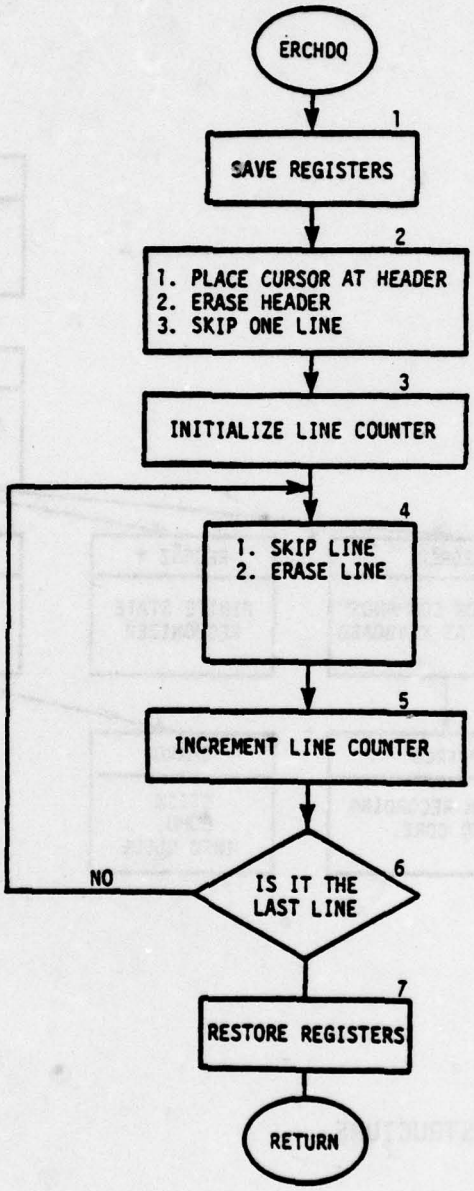
3.4 I/O Process

No addition or modifications were made in this process.

3.5 Off-Line Recording System

3.5.1 Functional Flow of Control. Shows the sequence of programs which perform the function described above on Chart 13.

CHART 12


ERCHDQ
1
SAVE REGISTERS
2
1. PLACE CURSOR AT HEADER
2. ERASE HEADER
3. SKIP ONE LINE
3
INITIALIZE LINE COUNTER
4
1. SKIP LINE
2. ERASE LINE
5
INCREMENT LINE COUNTER
6
IS IT THE LAST LINE
NO
7
RESTORE REGISTERS
RETURN

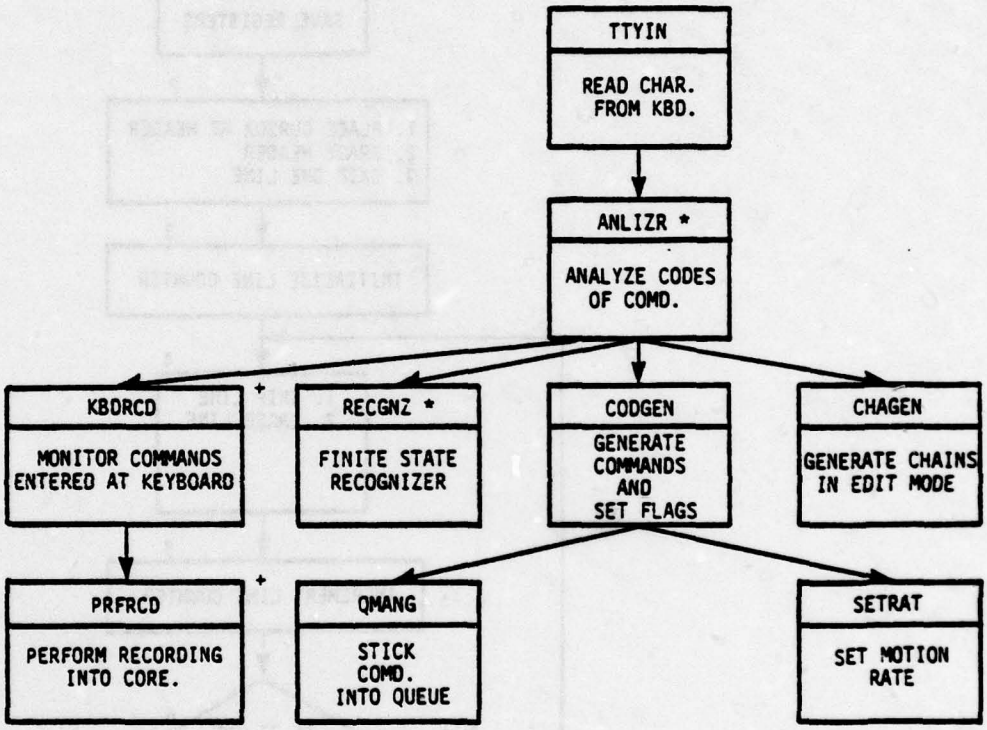
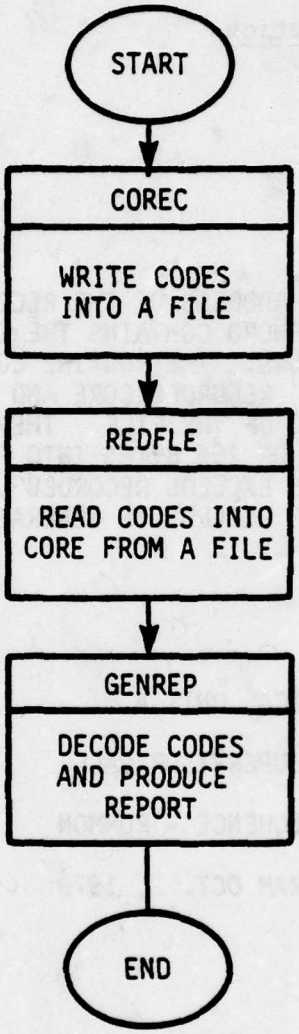


FIGURE 3-2.
TTY MODULE STRUCTURE

CHART 13


START
COREC
WRITE CODES INTO A FILE
REDFLE
READ CODES INTO CORE FROM A FILE
GENREP
DECODE CODES AND PRODUCE REPORT
END

3.5.2 Programs Documentation

NAME:

COREC

FUNCTION:

COPIES

ALGORITHM:

THE BEGINNING ADDRESS OF THE RECORDED CORE IS FIXED. THE FIRST HALFWORD CONTAINS THE END+1 BYTE ADDRESS OF RECORDED CORE. THE ROUTINE COMPUTES THE SIZE OF (NO. OF BYTES) RECORDED CORE AND STORES IT IN THE FIRST HALFWORD OF THE FILE. THEN IT CONTINUES TO WRITE RECORDS OF 256 BYTES INTO FILE UNTIL NO. OF RECORDED BYTES EXCEEDS RECORDED LENGTH OF CORE. AN INTEGER NO. OF RECORDS IS GENERATED.

PARAMETERS:

NONE

GLOBALS:

NONE

I/O:

WRITES TO LOGICAL UNIT 4.

CALLS:

SVC - SYSTEM SUPERVISOR CALL

CALLED BY:

JOB CONTROL SEQUENCE - RUNMON

HISTORY:

WRITTEN BY YORAM OCT. 2, 1978

NAME:

REDIFILE

FUNCTION:

READS A FILE OF RECORDED SESSION INTO SPECIFIED CONTINOUS AREA IN CORE.

ALGORITHM:

CONSECUTIVE RECORDS OF 256 BYTES ARE READ INTO SPECIFIED ADDRESSES IN CORE. TERMINATES WHEN ALL RECORDS OF FILE WERE STORED IN CORE.

PARAMETERS:

FIRST HALFWORD OF FILE CONTAINS NO. OF RECORDED BYTES.

RETURNS:

NONE

GLOBALS:

READS FROM LOGICAL UNIT 4.

I/O:

NONE

CALLS:

SVC - SUPERVISOR CALL TO READ FROM DISC.

CALLED BY:

JOB CONTROL SEQUENCE - RUNMON.

HISTORY:

WRITTEN BY YORAM OCT. 2 1978.

NAME:

GENREP

FUNCTION:

GENERATES MONITORING REPORT OF ONR78 SESSION EVENTS
WHICH ARE STORED IN CORE.

ALGORITHM:

1. READ DATA LENGTH WHICH STARTS IN FIXED PREDEFINED ADDRESS.
2. COMPUTE END ADR. AND SAVE IT.
3. PRINT REPORT HEADER USING FORTRAN SUBROUTINE.
4. GET NEXT CODE.

BEGIN ALTERNATIVE PROCESS HERE

5. CHECK IF NOT END DATA, IF SO TERMINATE.
6. COMPUTE TIME ELAPSED FROM SESSION BEGINNING.
7. PRINT IT IN A NEW LINE.
8. GET NEXT CODE.
9. FIND TYPE OF CODE
   0 - X'24' IS KEYBOARD
   X'30' - X'3F' IS EXECUTION
   X'40' - X'41' IS JOYSTICKS
   X'50' - X'51' IS CANCEL-ERROR
10. PRINT TYPE AND GOTO CODE PROCESSING SECTION.
11. PROCESS CODE BY EITHER: KBD OR EXEC OR JSTK OR CANERR UNTIL NEXT CODE NEGATIVE THEN GOTO 5A.

PARAMETERS:

NONE

RETURNS:

NONE

GLOBALS

NONE

I/O:

WRITES REPORT TO LOGIC UNIT NO. 3.

CALLS

SAVEME - FORTRAN SUBROUTINE WHICH PRINTS HEADER AND TIME IN EVERY NEW LINE.
PRNTEX - PRINT EXECUTION CODES.
GETCOD - INTERNAL ROUTINE TO GET NEXT CODE.

CALLED BY:

JOB CONTROL SEQUENCE - RUNMON

WRITTEN BY:

YORAM, DEC. 15, 1978

The meaning of hexacodes is shown in Appendix II.

PRNTEX -- DEC. 14, 1978 -- YORAM ALPEROVITCH

PRINT PRIMITIVE ROUTINE

GENERAL DESCRIPTION

THE ROUTINE WRITES A PRIMITIVE COMMAND UPON REQUEST OF A CALLING PROGRAM. POINTER TO THE PRIMITIVE LOCATION IS PASSED IN REGISTER 2. THE ROUTINE DECODES THE POINTED CODE BY COMPUTING REQUIRED TABLE ENTRY AND STORES ASCII CODES FROM THAT ENTRY INTO A WRITE BUFFER WHICH IS WRITTEN TO LOGICAL UNIT NO. 3.

LINKAGE

PRIMITIVE POINTER IS PASSED IN REGISTER 2.

CALLS

NONE

PRIMITIVES AND THE ASSOCIATED TABLE ENTRIES:

NAME	CODE	TABLE ENTRY
ROTATE LEFT	1 0	0*(CELSZE)
ROTATE RIGHT	1 1	1
GRASP	2 1	2
RELEASE	2 0	3
FORWARD	3 0	4
BACKWARD	3 1	5
MANUAL	7 0	6
GOTO POINT	4 X	7
GOTO PATH	5 X	8
GOTO REVERSE PATH	6 X	9
GOTO CHAIN	15 X	10

NAME:

SAVEME

FUNCTION:

1. PRINTS REPORT HEADER.
2. SKIPS TO A NEW LINE AND PRINTS TIME.

PARAMETERS:

TIME - INTEGER

RETURNS:

NONE

I/O:

WRITES TO LOGICAL UNIT 3.

CALLED BY:

GENREP

4. OPERATIONAL PROCEDURES

4.1 ONR78 Operation Sequences

4.1.1 Starting Sequence

At the Manipulator's Site:

(1) Turn on the hydraulic pump.

(2) Open the hydraulic valve on the manipulator.

(3) Turn on the video cameras.

At the Experimenting Site:

(1) Turn on the button for power, hydraulic and computer on the upper right hand side of the operator's keyboard.

(2) Turn on the two video displays.

(3) Turn on the feedback control display.

At the Computer Site:

(1) Turn on the +I/O.

(2) Turn on the main disk switch (with red color on it).

(3) Open the disk's case and put in the ONR78 disk.

(4) Make sure that both "write protect" are ON.

(5) Close the disk's case and press ON/OFF button.

(6) Turn on the ADDS terminal.

(7) Bring up the disk operating system, DOS, by the following sequence:

(a) Set rotary switch to ADR/MRD. (b) Set panel switches to HX'02D0'. (c) Reset RUN and SGL switches and press EXC switch. (d) If DOS doesn't show on screen--repeat Step a through c--if it still doesn't work--get help.

(8) Wait until the READY button is lit before proceeding with the rest of the sequence.

(9) Activate the system load module file by typing: AC ONR78,1C7.

(10) Load and start by typing: LO 1 and then ST 2000.

(11) Turn off the disk drive by pressing the ON/OFF switch.

(12) Start experimentation.

4.1.2 Stopping Sequence

At the Computer:

(1) Reset all panel switches--so that the system stops.

(2) Make sure the disks are not running or else turn them off and wait for the SAFE light before proceeding.

(3) Turn off the disc's case using the big red switch.

(4) Turn off the computer, terminal and +I/O.

At the Manipulator Site:

(1) Turn off hydraulic valve and pump.

(2) Turn off the video cameras.

At the Experimenting Site:

(1) Turn off the operator's keyboard.

(2) Turn off the video displays and control display.

4.2 Report Generation Operating Sequence

This sequence should be executed immediately after experimental session sequence.

4.2.1 Starting Sequence

At the Computer:

(1) Reset all panel switches so that system stops.

(2) Perform the usual stopping sequence at the manipulator and experimenting sites as shown in 4.1.2.

(3) Open the disc's case and put MOSHE disc in instead of the ONR78.

(4) Remove the "write protect" from the upper disc.

(5) Turn-on the disc driver.

(6) Turn-on the line printer.

(7) Wait until the READY button is lit and then proceed.

(8) Activate the system control file by typing AC RUNMON, 5C7.

(9) Transfer control to it by typing: TR 5.

(10) When report printer terminates perform the stopping sequence.

4.2.2 Stopping Sequence

(1) Turn off the discs by pressing the ON/OFF button.

(2) Turn off the computer, terminal and line printer.

(3) Wait until the SAFE light is on (very important!!)

(4) Turn off the main disc switch.

APPENDIX I

ONR78 - SESSION EVENTS MONITOR

TIME	EVENT	DETAILED DESCRIPTION		
15	KBD-CMD	STOP		
18	KBD-CMD	CONTINUE		
21	KBD-CMD	ROTATE	0	D 0
21	EXEC-CMD	ROTATE LEFT		
23	KBD-CMD	ROTATE	1	D 0
23	EXEC-CMD	ROTATE RIGHT		
36	KBD-CMD	STOP		
44	KBD-CMD	ROTATE	0	D 0
46	KBD-CMD	ROTATE	1	D 0
50	KBD-CMD	FORWARD	D 0	
52	KBD-CMD	BACKWARD	D 0	
53	KBD-CMD	GRASP	D 0	
55	KBD-CMD	RELEASE	D 0	
57	KBD-CMD	MANUAL	D 0	
60	KBD-CMD	CONTINUE		
60	EXEC-CMD	ROTATE LEFT		
72	EXEC-CMD	ROTATE RIGHT		
84	EXEC-CMD	FORWARD		

```
85
                EXEC-CMD    BACKWARD
86
                EXEC-CMD    GRASP
86
                EXEC-CMD    RELEASE
86
                EXEC-CMD    MANUAL
89
                KBD-CMD     CONTINUE
1397
                KBD-CMD      DEFINE     CHAIN      1      D 0
1415
                KBD-CMD     MANUAL      D 0
1422
                KBD-CMD      DO TO      CHAIN      1      D 0
1426
                KBD-CMD      END        CHAIN     D 0
1435
                KBD-CMD      GO TO      CHAIN     D 0
1435
                EXEC-CMD    MANUAL
1438
                KBD-CMD     CONTINUE
1438
```

APPENDIX II

Recorded Hexa-Codes Meaning

(1) Keyboard input hexadecimal codes.

0	-	sensor
1	-	spatial control
2	-	joint control
3	-	frozen wrist
4	-	motion rate
5	-	skip command
6	-	clear command
7	-	initialize system
8	-	goto
9	-	define
A	-	delete
B	-	end
C	-	forward
D	-	backward
E	-	grasp
F	-	release
10	-	manual
11	-	rotate
12	-	continue
13	-	stop
14	-	0
15	-	1
16	-	2
17	-	3
18	-	4
19	-	5
1A	-	6
1B	-	7
1C	-	8
1D	-	9
1E	-	point
1F	-	path
20	-	reverse path
21	-	chain

22 - do
23 - do now
24 - cancel

(2) Execution codes.

31 - rotate (left, right)
32 - grip (grasp, release)
33 - forward, backward
34 - goto point
35 - goto path
36 - goto rev. path
37 - manual
3F - goto chain

(3) Joystick Codes.

40 - start operating
41 - end operating

(4) Special Codes.

50 - cancel
51 - error

DISTRIBUTION LIST

Director, Engineering Psychology
Programs, Code 455
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Defense Documentation Center
Cameron Station
Alexandria, VA 22314

Dr. Stephen J. Andriole
Cybernetics Technology Office
Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

CDR. Paul R. Chatelier
Military Asst. for Training
and Personnel Technology
Under Secretary of Defense Office
OUSDRE (EL&S), Pentagon
Washington, D.C. 20301

Director, Information Systems
Program, Code 437
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Program Director, Ocean Technology
Naval Ocean Research & Development
(NORDA)
Bay St. Louis, Mississippi 39529

Commanding Officer
ONR Branch Office
Attn: Mr. R. Lawson
1030 East Green Street
Pasadena, CA 91106

Director, Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375

Office of the Chief of Naval
Operations, OP987H
Personnel Logistics Plans
Department of the Navy
Washington, D.C. 20350

Dr. Andreas B. Rechnitzer
Office of the Chief of Naval
Operations
Oceanography Division OP-952
Washington, D.C. 20350

Mr. Arnold Rubenstein
Naval Material Command
NAVMAT 08T24
Department of the Navy
Washington, D.C. 20360

Mr. Glen Spalding
Naval Material Command,
MAT 08T24
Crystal Plaza
Washington, D.C. 20360

Commander, Naval Facilities
Engineering Command
R&D Plans & Programs Division
Code 031A
Alexandria, VA 22332

Director
Behavioral Sciences Department
Naval Medical Research Institute
Bethesda, M.D. 20014

Dr. George Moeller
Human Factors Engineering Branch
Submarine Medical Research Laboratory
Naval Submarine Base
Groton, CT 06340

Mr. Phillip Andrews
Naval Sea Systems Command
NAVSEA 0341
Washington, D.C. 20362

Dr. Bruce Wald
Communications Sciences Division
Code 7500
Naval Research Laboratory
Washington, D.C. 20375

Dr. John K. Dixon
Computer Scientist
Naval Research Laboratory
Washington, D.C. 20375

Dr. John Silva
Man-System Interaction Division
Code 823, Naval Ocean Systems Center
San Diego, CA 92152

Mr. John Quirk
Naval Coastal Systems Laboratory
Code 712
Panama City, FL 32401

Human Factors Department
Code N215
Naval Training Equipment Center
Orlando, FL 32813

Dr. Gary Poock
Operations Research Department
Naval Postgraduate School
Monterey, CA 93940

Dr. A.L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Technical Director
U.S. Army Human Engineering Labs
Aberdeen Proving Ground
Aberdeen, MD 21005

U.S. Air Force Office of Scientific Research
Life Sciences Directorate, NL
Bolling Air Force Base
Washington, D.C. 20332

Lt. Col. Joseph A. Birt
Human Engineering Division
Aerospace Medical Research Laboratory
Wright Patterson AFB, OH 45433

Dr. W.S. VAughan
Oceanautics, Inc.
422 6th Street
Annapolis, MD 21403

Dr. Ross L. Pepper
Naval Ocean Systems Center
Hawaii Laboratory
P.O. Box 997
Kailua, Hawaii 96734

Mr. Jim Katayama
Naval Ocean Systems Center
Hawaii Laboratory
P.O. Box 997
Kailua, Hawaii 96734

CDR Thomas Berghage
Naval Health Research Center
San Diego, California 92138

Mr. W. Greenert
Naval Material Command
NAVMAT 034
Hoffman II Building
200 Stovall Street
Alexandria, VA 22332

Mr. Paul Heckman
Naval Ocean Systems Center
San Diego, CA 92152

Dr. J. Miller
National Oceanic and Atmospheric
Administration
11400 Rockville Pike
Rockville, M.D. 20852

Dr. W. Mehuron
Office of the Chief of Naval
Operations
OP 009T
Washington, D.C. 20350

Mr. H. Talkington
Ocean Engineering Department
Naval Ocean Systems Center
San Diego, CA 92152

Dr. T.B. Sheridan
Department of Mechanical
Engineering
Massachusetts Institute of
Technology
Cambridge, MA 02139

Dr. A. Baggeroer, Assoc. Prof.
of Ocean & Electrical Eng.
Room 5-326
Massachusetts Inst. of Technology
Cambridge, MA 02139

Mr. R. Wernli
Naval Ocean Systems Center
Ocean Technology Department
San Diego, CA 92152

Mr. J. Williams
Department of Environmental
Sciences
U.S. Naval Academy
Annapolis, M.D. 21402

Dr. William L. Verplank
Xerox Vusiness Systems
Systems Development Department
701 So. Aviation Blvd.
El Segundo, CA 90245

Mr. Clifford Winget
Woods Hole Oceanographic Inst.
Woods Hole, MD 02543